Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра дифференциальных и стохастических уравнений

РАБОТА ПРОВЕРЕНА
Рецензент, доцент кафедры комп. топологии и алгебры ЧелГУ,
канд. физ.-мат. наук, доцент
_____ / О.В. Митина /
«03» июня 2016 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой дифференциальных и стохастических уравнений, ЮУрГУ,
доктор физ-мат. наук, доцент
_____ / С.А. Загребина /
«___» _____ 2016 г.

НЕПРИВОДИМЫЕ МНОГОЧЛЕНЫ НАД КОНЕЧНЫМИ ПОЛЯМИ И ИХ ПРИМЕНЕНИЕ В КРИПТОГРАФИИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2016.129-129.ВКР

Руководитель, доктор физ.-мат. наук, доцент
_____ / Н.Д. Зюляркина /
«07» июня 2016 г.

Автор, студент группы ММиКН-293
_____ / А.В. Корсун /
«02» июня 2016 г.

Нормоконтролер, канд. физ.-мат. наук, доцент
_____ / М.А. Сагадеева /
«___» _____ 2016 г.

Челябинск 2016

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра дифференциальных и стохастических уравнений

З А Д А Н И Е

студенту группы ММиКН-293
Корсуну Александру Валерьевичу
на выпускную квалификационную работу
по направлению 09.04.04 – ПРОГРАММНАЯ ИНЖЕНЕРИЯ

1. **Тема диссертации:** «Неприводимые многочлены над конечными полями и их применение в криптографии».
   (Утверждена приказом по университету от «15» апреля 2016г. № 661)
2. **Перечень подлежащих исследованию вопросов**
   2.1. Литературный обзор
   2.2. Постановка задачи
   2.3. Алгоритм факторизации многочленов над конечным полем (алгоритм Берлекэмпа)
   2.4. Изучение строения факторкольца кольца многочленов
   2.5. Разработка криптосистемы на основе факторкольца кольца многочленов
   2.6. Генерация неприводимых многочленов над конечным полем
3. **Календарный план подготовки выпускной квалификационной работы**

Наименование этапов дипломной работы	Срок выполнения этапов работы	Отметка о выполнении
1. Обзор литературы и постановка задачи	01.02.16 – 14.02.16	Зюл
2. Анализ строения факторкольца кольца многочленов	15.02.16 – 28.02.15	Зюл
3. Разработка криптосистемы на основе факторкольца кольца многочленов	29.02.16 – 13.03.16	Зюл
4. Разработка и тестирование программных средств	14.03.16 – 17.04.16	Зюл
5. Подготовка текста выпускной квалификационной работы	18.04.16 – 15.05.16	Зюл
6. Проверка и рецензирование работы руководителем, исправление замечаний	16.05.16 – 25.05.16	Зюл
7. Подготовка доклада и текста выступления	26.05.16 – 10.06.16	Зюл
8. Внешнее рецензирование	20.05.16 – 05.06.16	
9. Защита дипломной работы	10.06.16 – 20.06.16	

4. **Дата выдачи задания** «01» февраля 2016 г.

Руководитель работы
доктор ф.- м. наук, доцент _____ (подпись) ***Зюляркина Н.Д.***

Задание принял к исполнению _____ (подпись) ***Корсун А.В.***

УДК 512.624.2

Корсун А.В.

Неприводимые многочлены над конечными полями и их применение в криптографии. /А.В. Корсун. – Челябинск, 2016. - 23 с.

Квалификационная работа посвящена исследованию неприводимых многочленов над конечным полем, генерации неприводимых многочленов, а также описанию и разработке криптосистемы на основе факторкольца многочленов.

Библиографический список – 18 наим.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Неприводимые многочлены, корни которых образуют базис для представления элементов конечных полей, аналогичны простым числам [6]. Они нашли свое применение в различных областях математики, информационной техники и защите информации. Использование свойств неприводимых многочленов [12] позволяет максимизировать эффективность компьютерной реализации арифметики в конечных полях, что имеет особое значение для криптографии и теории кодирования. Таковы, например, реализация электронной цифровой подписи на эллиптических кривых [2]; коды Рида-Маллера, Рида-Соломона и другие; программирование дискретных устройств [10, 16]; современные стандарты шифрования.

Для нахождения таких многочленов нет определенного (детерминированного) алгоритма, так что их построение производится подбором, т.е. вероятностными алгоритмами, что требует временных затрат и объемных вычислений. Генерация неприводимых многочленов является актуальной и сложной на сегодняшний день прикладной задачей, широко используемой и востребованной в криптографических приложениях (генерация открытых и закрытых ключей) и теории кодирования (построение кодов полиномиального кодирования).

Генерация неприводимых многочленной есть порождение (нахождение) неприводимых многочленов с заведомо хорошими свойствами, что позволяет использовать их для генерации ключей (закрытых и открытых) в системах защищенного документооборота, например, в системах интернет-банка и клиент-банка. Коэффициенты многочленов, как элементы конечных полей характеристики два, можно интерпретировать как многобитовые последовательности для передачи по современным каналам связи. Поэтому важной задачей становится использование неприводимых многочленов больших степеней, задающих эти поля.

Целью работы является исследование области применения неприводимых многочленов над конечными полями. Для достижения этой цели необходимо:

1. Изучить и реализовать, в системе компьютерной алгебры GAP, алгоритм факторизации многочленов над конечным полем (алгоритм Берлекэмпа).
2. Изучить строение факторкольца кольца многочленов над конечным полем.

3. Разработать и реализовать, в системе компьютерной алгебры GAP, криптосистему с открытым ключом на основе факторкольца кольца многочленов над конечным полем.
4. Изучить и реализовать, в системе компьютерной алгебры GAP, рекурсивный алгоритм генерации неприводимых многочленов над конечным полем.

1. ФАКТОРИЗАЦИЯ МНОГОЧЛЕНОВ

1.1. Описание задачи

Любой многочлен положительной степени над заданным полем можно разложить в произведение неприводимых многочленов. Если рассматриваемое поле конечно, то для фактического вычисления неприводимых сомножителей данного многочлена положительной степени над этим полем существуют весьма эффективные алгоритмы.

Наличие таких алгоритмов для многочленов над конечными полями особенно важно для теории кодирования и для изучения линейных рекуррентных соотношений в конечных полях. Эти многочлены могут быть использованы для построения элементов большого порядка в матричных группах. Но и вне области конечных полей в алгебре и теории чисел имеется много вычислительных задач, которые так или иначе связаны с разложением многочленов над конечными полями. В этой связи можно упомянуть, например, разложение на множители многочленов над кольцом целых чисел, отыскание разложений простых рациональных чисел в полях алгебраических чисел, вычисление группы Галуа некоторого уравнения над полем рациональных чисел и построение расширений полей.

Первый алгоритм разложения многочлена на неприводимые множители с целыми коэффициентами за конечное число шагов был предложен австрийским астрономом Шубертом в 1793 году. Первый признак неприводимости многочлена над полем рациональных чисел предложен Эйзенштейном в 1850 году. В 1882 году Кронекер заново открыл метод Шуберта, а также предложил алгоритмы факторизации многочленов с двумя и белее переменными. В 1954 году Батлер предложил тест на неприводимость, основанный на построении «Q – матрицы». Первая достаточно эффективная в вычислительном плане компьютерная реализация алгоритма Кронекера для разложения на множители многочленов с коэффициентами из конечного поля была осуществлена в 1966. С этого момента начинается основной этап в построении быстрых алгоритмов факторизации многочленов над конечными полями. Настоящий прорыв был сделан Берлекэмпом в 1967 году. Им был предложен алгоритм, выполняющий разложение за полиномиальное время, с использованием матрицы Батлера. Ряд усовершенствований метода Берлекэмпа был выполнен в работах. Другой подход, основанный на нахождении корней алгебраических расширений и применением вероятностного анализа, разработан

Рабином в 1980 году. На сегодняшний день развиты эффективные вероятностные и детерминированные алгоритмы, имеющие полиномиальную сложность.

1.2. Алгоритм Берлекэмпа

Идея данного алгоритма [4] основана на китайской теореме об остатках для многочленов. Можно показать, что будет справедливо следующее утверждение.

Утверждение 1. Пусть $p_1(x), \dots, p_r(x)$ – многочлены из $Z_p[x]$, НОД $\left(p_j(x), p_k(x)\right) = 1$ при $j \neq k$. Допустим что, $s_1(x), \dots, s_r(x)$ – произвольные многочлены из $Z_p[x]$. Тогда существует единственный многочлен $t(x) \in Z_p[x]$, такой что $\deg(t(x)) < \deg\left(p_1(x)\right) + \cdots + \deg\left(p_r(x)\right)$ и $t(x) \equiv s_i(x)(mod\ [p_i(x)]), 1 \leq i \leq r$.

Теорема 1. В поле F_q порядка q, где q – примарное число имеет место разложение:

$$x^q - x = \prod_{s \in F_q} (x - s).$$

Доказательство 1. Обозначим через $f(x) = x^q - x$. Очевидно, что 0 – корень $f(x)$. Пусть $\alpha \in Z_q, \alpha \neq 0$. Так как F_q^* – циклическая группа порядка $q-1$. То $\alpha^{q-1} = 1 \Rightarrow \alpha^q = \alpha \Rightarrow f(\alpha) = 0$. То есть все элементы поля F_q являются корнями $f(x)$, и $\deg f(x) = |F_q| \Rightarrow f(x)$ раскрывается на линейные множители, то есть имеет место требуемое равенство.

Теорема 2. Пусть $p(x)$ и $t(x)$ – два нормированных многочлена над F_p, такие что

$$t(x)^p \equiv t(x)\left(mod\ p(x)\right), \deg\left(t(x)\right) < \deg\left(p(x)\right).$$

Тогда

$$p(x) = \prod_{s \in F_p} \text{НОД}(p(x), t(x) - s).$$

Доказательство 2. Из предположения следует, что $p(x) | t(x)^p - t(x)$. Поэтому

$$p(x) = \text{НОД}\left(p(x), t(x)^p - t(x)\right) = \text{НОД}(p(x), \prod_{s \in F_p} (t(x) - s).$$

Помимо этого $\text{НОД}(t(x) - s, t(x) - t) = 1$ для $s \neq t$, и многочлены $\text{НОД}(p(x), t(x) - s)$ и $\text{НОД}(p(x), t(x) - t)$ также взаимно просты. Поэтому

$$\text{НОД}(p(x), \prod_{s \in F_p} (t(x) - s) = \prod_{s \in F_p} \text{НОД}(p(x), t(x) - s).$$

Таким образом, пусть $p(x)$ – свободный от квадратов многочлен степени n, который нужно разложить на множители над F_p, и предположим, удалось найти многочлены $t(x) \in Z_p[x]$, $1 \leq \deg t(x) \leq n$, такие что $t(x)^p \equiv t(x)(mod\ p(x))$. По теореме, многочлен $x^p - x$

имеет в $Z_p[x]$ ровно p корней. А именно $0,1,\dots,p-1$. Значит он раскладывается следующим образом $x^p - x = x(x-1)(x-2)\dots(x-p+1)$. Заменив x на $t(x)$, в кольце $Z_p[x]$ получим $t(x)^p - t(x) = t(x)(t(x)-1)(t(x)-2)\dots(t(x)-p+1)$. Так как $p(x)|t(x)^p - t(x)$, то $p(x) = \text{НОД}(p(x), t(x)^p - t(x))$. Кроме того поскольку многочлены $t(x)-s$ и $t(x)-t$ – взаимно простые при $s \neq t$, то

$$p(x) = \text{НОД}\big(p(x), t(x)^p - t(x)\big) = \prod_{s \in F_p} \text{НОД}(p(x), t(x) - s)$$

– нетривиальное разложение многочлена над F_p.

Теперь задача состоит в определении многочленов $t(x)$. Это можно осуществить с помощью решения систем линейных уравнений, получаемой следующим образом. Пусть $t(x) = t_0 + t_1 x + \dots + t_{n-1}x^{n-1}$ – многочлен, где требуется найти коэффициенты. Нужно сначала проверить делит ли $p(x)$ многочлен $t(x)^p - t(x)$. Ранее доказано, что $t(x)^p = t_0 + t_1 x^p + \dots + t_{n-1}x^{(n-1)p}$.

Разделив x^{ip} на $p(x)$ получаем $x^{ip} = p(x)\cdot q_i(x) + r_i(x)$, где $r_i(x) = r_{i,0} + r_{i,1}x + \dots + r_{i,n-1}x^{n-1}$. Теперь, заменив x^{ip} на соответствующие выражения, получим $t(x)^p = t_0 r_0(x) + t_1 r_1(x) + \dots + t_{n-1}r_{n-1}(x) + [\text{кратное } p(x)]$.

Таким образом $p(x)|t(x)^p - t(x)$ тогда и только тогда когда $p(x)$ делит многочлен $t_0 r_0(x) + t_1 r_1(x) + \dots + t_{n-1}r_{n-1}(x) - (t_0 + t_1 x + \dots + t_{n-1}x^{n-1}) = t_0(r_0(x)-1) + \dots + t_{n-1}(r_{n-1}(x) - x^{n-1})$, степень которого $\leq n-1$. Поэтому многочлен степени n будет делить этот многочлен если только он равен нулю. Приравняв его к нулю и собрав коэффициенты при степенях x, получаем систему из n линейных уравнений $t_0, t_1, \dots, t_{n-1}$. Это и есть коэффициенты того многочлена $t(x)$.

Пусть $Q = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdot & \cdot & \cdot & r_{0,n-1} \\ r_{1,0} & r_{1,1} & \cdot & \cdot & \cdot & r_{1,n-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{n-1,0} & r_{n-1,1} & \cdot & \cdot & \cdot & r_{n-1,n-1} \end{pmatrix}$ – матрица, строки которой образуют коэффициенты многочленов остатков. По этому имеет место следующая теорема.

Теорема 3. Многочлен $t(x) = t_0 + t_1(x) + \dots + t_{n-1}x^{n-1}$ является решением сравнения $t(x)^p \equiv t(x)(mod\, p(x))$ тогда и только тогда, когда $(t_0, t_1, \dots, t_{n-1})(Q - E) = (0,0,\dots,0)$.

Пусть N – множество решений системы линейных однородных уравнений с матрицей $(Q-E)^t$. Данное множество является конечномерным векторным пространством.

Теорема 4. Число различных неприводимых сомножителей $p_i(x)$ многочлена $p(x)$ в $Z_p[x]$ равно размерности пространства N.

Доказательство 3. Многочлен

$$p(x)|\prod_{s\in F_p} \text{НОД}(p(x), t(x)-s),$$

тогда и только тогда, когда каждый $p_i(x)|t(x)-s_i$, $s_i \in Z_p$. По ранее доказанным фактам для набора s_i существует единственный многочлен $t(x)\big(mod\, p(x)\big)$, такой что $s_i(x) \equiv t(x)(mod\, p_i(x))$. Существует p^r решений сравнения $t(x)^p - t(x) \equiv 0(mod\, p(x))$. $t(x)$ является решением сравнения если $(t_0, t_1, \dots, t_{n-1})Q = (t_0, t_1, \dots, t_{n-1})$.

Теорема 5. Многочлен $p(x)$ степени $n > 1$ неприводим в $Z_p[x]$ тогда и только тогда, когда пространство N одномерно и $\text{НОД}\big(p(x), p'(x)\big) = 1$.

Доказательство 4. Пространство N одномерно тогда и только тогда, когда $p(x) = p_1(x)^k$ – степень неприводимого многочлена. Тогда берём $r(x) = 1$.

Теорема 6. Пусть $p(x) = p_1(x) \cdot \dots \cdot p_r(x)$ в $Z_p[x]$ и $\{b_1, \dots, b_r\}$ – базис нуль-пространства. Тогда для каждого $j' \neq j$, $1 \leq j < j' \leq r$, существует k $1 \leq k \leq r$ и $s \in Z_p$, такие что $p_j(x)$ делит, а $p_{j'}(x)$ не делит $\text{НОД}(p(x), b_k(x) - s)$.

Доказательство 5. В нуль-пространстве существует вектор, -ая компонента которой отлична от j'-ой. Значит найдётся такое k, $1 \leq k \leq r$, $b_k(x)(mod\, p_j(x)) \neq b_k(x)(mod\, p_{j'}(x))$. Положим $b_k(x)\left(mod\, p_j(x)\right) = s \in Z_p$.

Вход: Нормированный, свободный от квадратов многочлен $p(x) \in F_p$, $\deg[p(x)] = n$.

Выход: Неприводимые над F_p сомножители многочлена $p(x)$.

Описание реализации:

1. Построить матрицу Q.

2. Найти фундаментальную систему решений системы линейных однородных уравнений с матрицей $(Q-E)^t$ и записать соответствующий данной фундаментальной системе решений многочлен.

3. Вычисление сомножителей. Пусть $b_2(x)$ – многочлен, соответствующий вектору b_2. Вычислим $\text{НОД}(p(x), b_2(x) - s)$ для всех $s \in F(p)$. Если с помощью $b_2(x)$ получено менее r сомножителей, вычислим $\text{НОД}(\omega(x), b_2(x) - s)$ для всех $s \in F(p)$ и всех сомножителей $\omega(x)$, найденных к данному времени, $k = 3, 4, \dots, r$, пока не найдётся r сомножителей.

На шаге 2 этого алгоритма матрица Q приводится к предпочтительному виду, затрачивается время $O(n^3)$. Так как требуется не более p вычислений НОД для каждого

базисного вектора и не более r из этих вычислений будут нетривиальны, то $t_{BA}(t(x)) = O(prn^3)$. Так что алгоритм не очень эффективен при больших p.

Пример. Разложим над F_{13} многочлен $p(x) = 8x^4 + 6x^3 + 8x^2 + 3x + 12$, свободный от квадратов.

Решение. Вместо данного многочлена рассмотрим нормированный эквивалентный многочлен $x^4 + 4x^3 + x^2 + 2x + 8$.

Для начала вычислим обратные элементы ненулевым элементам F_{13} $(1, \dots, 12)$. Это соответственно будут (1,7,9,10,8,11,2,5,3,4,6,12).

Первая строка матрицы Q [4x4] всегда представляет собой (1,0,0,0), соответствуя многочлену $x^0(mod\, p(x)) = 1$. Вторая строка представляет $x^{13}(mod\, p(x))$, третья $x^{26}(mod\, p(x))$, четвёртая $x^{39}(mod\, p(x))$.

Пусть $p(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x + c_0$. Предположим, что $x^k \equiv r_{k,n-1}x^{n-1} + \cdots + r_{k,1}x + r_{k,0}(mod\, p(x))$. Тогда $x^{k+1} \equiv (r_{k,n-1}x^n + \cdots + r_{k,1}x^2 + r_{k,0}x)$ или $x^{k+1} \equiv r_{k,n-1}(-c_{n-1}x^{n-1} - \cdots - c_1x - c_0) + r_{k,n-2}x^{n-1} + \cdots + r_{k,1}x^2 + r_{k,0}x$.

Что означает $x^{k+1} \equiv r_{k+1,n-1}x^{n-1} + \cdots + r_{k+1,1}x + r_{k+1,0}\,(mod\, p(x))$. Здесь $r_{k+1,j} = r_{k,j-1} + r_{k,n-1}c_j$, $r_{k,-1} = 0$.

Эти формулы объясняют вычисление $x^i(mod\, p(x))$. Вычисления можно проводить используя массив $r_{n-1}, \dots, r_1, r_0$. В цикле $t := r_{n-1}$, $r_{n-1} := (r_{n-2} - tc_{n-1})(mod\, p)$, … , $r_1 := (r_0 - tc_1)(mod\, p)$, $r_0 := (-tc_0)(mod\, p)$. Результаты отображаем в таблице:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$r_{k,0}$	1	0	0	0	5	6	10	9	6	3	1	0	10	12
$r_{k,1}$	0	1	0	0	11	0	2	9	4	9	0	1	9	0
$r_{k,2}$	0	0	1	0	12	2	11	8	0	6	2	0	12	4
$r_{k,3}$	0	0	0	1	9	2	7	9	11	8	0	2	5	5

Нетрудно видеть вторую строку матрицы Q: (12,0,4,5). Аналогично строим для $k = 26, 39$ и получаем матрицу

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 12 & 0 & 4 & 5 \\ 12 & 5 & 6 & 12 \\ 1 & 4 & 3 & 6 \end{pmatrix}, Q - E = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 12 & 12 & 4 & 5 \\ 12 & 5 & 5 & 12 \\ 1 & 4 & 3 & 5 \end{pmatrix}.$$

Теперь нужно находить нуль-пространство матрицы $Q - I$. На основании эквивалентных преобразований матрицы составляется следующий алгоритм:

Вход: Матрица размера n $M = (m_{i,j}), 0 \le i,j \le n-1$, с элементами из поля.

Выход: Линейно независимые вектора $b_1, \dots, b_r$, такие что $b_i M = 0$, $n - r$ – ранг матрицы M.

Реализация:

1. $r := 0; c_0 := c_1, \dots, c_{n-1} := -1$.

2. Для h от 0 до $n-1$: если найдётся столбец с номером h и $m_{h,j} \neq 0$, $c_j < 0, j = 0, \dots, n-1$, то j столбец матрицы M умножаем на $\frac{-1}{m_{h,j}}$, чтобы $m_{h,j} = -1$, затем для всех $i \neq j$ прибавляем умноженный на $m_{h,j}$ столбец j к столбцу i. И $c_j := h$. Если не найдётся столбца j, чтобы $m_{h,j} \neq 0$, то положить $r = r + 1$, выдать вектор $b_r = (b_{r,0}, b_{r,1}, \dots, b_{r,n-1})$, где для $j = 0,1, \dots, n-1$.

$$h_{r,j} := \begin{cases} m_{hk} & \text{если, } c_k = j > 0, \text{если таких } k \text{ не одно, то взять любое,} \\ 1 & \text{если, } j = h, \\ 0 & \text{в противном случае.} \end{cases}$$

При $h = 0$ получится вектор $b_1 = (1,0, \dots,0)$. Он соответствует многочлену-константе 1. При $h = 1$ можно взять j равным 0,1,2,3, поскольку $c_i = -1$ для $i = 1,2,3$ – выбор на данном этапе полностью произволен, хотя он и влияет на получаемые при выходе векторы. Берём $j = 0$ и после ранее описанных преобразований матрица Q имеет вид:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 \\ 12 & 6 & 1 & 7 \\ 1 & 3 & 7 & 10 \end{pmatrix}.$$

Второй элемент в первом столбце 12 – означает $c_0 = 1$. Для $h = 2$ матрица будет

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 8 & 6 & 0 & 0 \end{pmatrix}.$$

Третий элемент второго столбца означает, что $c_1 = 2$. Два последние столбца, состоящие только из нулей, обуславливают на выходе вектор $b_2 = (0,8,6,1)$ при $h = 3$. Соответствующий многочлен будет $x^3 + 6x^2 + 8x$.

Из вида матрицы $Q - E$ при $h = 3$ видно, что векторы b_1 и b_2 удовлетворяют условию $b_i M = 0$. Так как эти вычисления дали только два линейно независимых вектора, то $p(x)$ должен иметь только два неприводимых сомножителя над F_{13}.

Теперь нужно переходить к третьему шагу алгоритма Берлекэмпа, в котором непосредственно найдутся эти сомножители. Этот шаг состоит в нахождении НОД$(p(x), b_2(x) - s)$ для всех $s \in F_{13}$. Здесь $p(x) = 8x^4 + 6x^3 + 8x^2 + 3x + 12$ и $b_2(x) = x^3 + 6x^2 + 8x$. После вычислений получаем при $s = 1$ НОД$(p(x), b_2(x) - 1) = x^2 + 9x + 9$ и при $s = 9$ НОД$(p(x), b_2(x) - 9) = x^2 + 8x + 11$. Непосредственная проверка показывает, что многочлены найдены правильно.

2. ФАКТОРКОЛЬЦО КОЛЬЦА МНОГОЧЛЕНОВ

Определение 1. Пусть K – кольцо, I – подкольцо кольца K. Подкольцо I называется идеалом кольца K, если выполняются следующие условия:

$$\forall a \in K, a \cdot I \subseteq I; \tag{1}$$

$$\forall a \in K, I \cdot a \subseteq I. \tag{2}$$

Замечание. Если K – коммутативное кольцо, то условия (1) и (2) эквивалентны.

Определение 2. Пусть K – кольцо, I – идеал. Обозначим через K/I – множество всех правых (левых) смежных классов K по I. Определим на K/I сложение и умножение по следующим правилам:

$$(a + I) + (b + I) = (a + b) + I, \tag{3}$$

$$(a + I) \cdot (b + I) = (a \cdot b) + I. \tag{4}$$

Тогда $(K/I, +, *)$ – называется факторкольцом кольца K по идеалу I.

Легко доказать, что операции (3) и (4) определены корректно, то есть не зависят от выбора конкретного представителя a класса смежности $a + I$. Например, пусть $a_2 = a + i_1,\ b_2 = b + i_2,\ i_{1,2} \in I$. Тогда $a_2 \cdot b_2 = (a + i_1) \cdot (b + i_2) = a \cdot b + i_1 \cdot b + a \cdot i_2 + i_1 \cdot i_2 \in a \cdot b + I$.

Определение 3. Пусть $K = P[x]$ – кольцо многочленов, где P – поле. Зафиксируем $f(x) \in K$ и рассмотрим $I_{f(x)} = \{g(x) \in K, h(x) \in K \mid g(x) = h(x) \cdot f(x)\}$. Тогда легко показать, что $I_{f(x)}$ – идеал K. Данный идеал называется главным идеалом, порожденный многочленом $f(x)$. Случай, когда $f(x) = C$, при $C = 0$ или $C \neq 0$ является вырожденным. То есть при $f(x) = 0 \Rightarrow K/I_{f(x)} \simeq K$, а при $f(x) = C \neq 0 \Rightarrow K/I_{f(x)} = \{0\}$. Для удобства можно представить $K/I_{f(x)}$ в виде $\{g(\alpha) \in P[\alpha] \mid \deg g(\alpha) < \deg f(x)\}$.

Замечание. Элементы $K/I_{f(x)}$ можно представить в виде $g(x) + I$, где $\deg g(x) < \deg f(x)$.

Утверждение 1. Если $f(x)$ – неприводимый многочлен над K, то $K/I_{f(x)}$ является полем.

Утверждение 2. Пусть F_q – конечное поле порядка q, где q – примарное число и $f(x), g(x) \in F_q[x]$ – неприводимые многочлены, $\deg f(x) = m, \deg g(x) = n$. Наибольший общий делитель данных многочленов равен 1, из-за того, что оба многочлена неприводимы над F_q. Пусть $K = F_q[x]/(f(x) \cdot g(x))$ – факторкольцо кольца многочленов

$F_q[x]$. Тогда порядок группы $|K^*| = (q^m - 1) \cdot (q^n - 1)$. Данный порядок группы является количеством обратимых многочленов в факторкольце K.

Доказательство 1. Рассмотрим кольцо K и многочлен $h(\alpha) \in K$. Поделим $h(\alpha)$ на $g(\alpha)$ и $f(\alpha)$ с остатком. Получим:

$$\begin{cases} h(\alpha) \equiv h_1(\alpha) (mod\ f(\alpha)), \\ h(\alpha) \equiv h_2(\alpha) (mod\ g(\alpha)). \end{cases} \quad (5)$$

Покажем, что $\forall\, h_1(\alpha)$ и $h_2(\alpha)$, существует единственный многочлен $h(\alpha) \in K$ для которого выполняется условие (5). Всего элементов в K будет q^{m+n}. Так как $h_1(\alpha)$ можно выбрать q^m способов, а $h_2(\alpha)$ можно выбрать q^n способов. Поэтому различных наборов $(h_1(\alpha), h_2(\alpha))$ существует в точности $q^m \cdot q^n = q^{m+n}$. Тогда количество наборов равно количеству элементов кольца K. И различным элементам кольца K соответствуют различные наборы, следовательно каждому элементу $h(\alpha) \in K$ соответствует единственный набор вида $(h_1(\alpha), h_2(\alpha))$ и наоборот. Заметим, что

$$h(\alpha) \in K^* \Leftrightarrow \begin{cases} НОД(h(\alpha), f(\alpha)) = 1, \\ НОД(h(\alpha), g(\alpha)) = 1. \end{cases} \quad (6)$$

То есть

$$\begin{cases} h(\alpha) \equiv h_1(\alpha) (mod\ f(\alpha)), h_1(\alpha) \not\equiv 0, \\ h(\alpha) \equiv h_2(\alpha) (mod\ g(\alpha)), h_2(\alpha) \not\equiv 0. \end{cases} \quad (7)$$

Способов выбрать $h_1(\alpha)$ будет $q^m - 1$, соответственно способов выбрать $h_2(\alpha)$ будет $q^n - 1$. Поэтому $|K^*| = (q^m - 1) \cdot (q^n - 1)$.

3. КРИПТОСИСТЕМА НА ОСНОВЕ НЕПРИВОДИМЫХ МНОГОЧЛЕНОВ

3.1. Определение криптосистемы с открытым ключом

Главная проблема использования одноключевых (симметричных) криптосистем заключается в распределении ключей. Для того, чтобы был возможен обмен информацией между двумя сторонами, ключ должен быть сгенерирован одной из них, а затем в конфиденциальном порядке передан другой. Особую остроту данная проблема приобрела в наши дни, когда криптография стала общедоступной, вследствие чего количество пользователей больших криптосистем может исчисляться сотнями и тысячами.

Начало асимметричным шифрам было положено в работе «Новые направления в современной криптографии» Уитфилда Диффи и Мартина Хеллмана, опубликованной в 1976 г. Находясь под влиянием работы Ральфа Меркле (Ralph Merkle) о распространении открытого ключа, они предложили метод получения секретных ключей для симметричного шифрования, используя открытый канал. В 2002 г. Хеллман предложил называть данный алгоритм «Диффи - Хеллмана - Меркле», признавая вклад Меркле в изобретение криптографии с открытым ключом.

Хотя работа Диффи-Хеллмана создала большой теоретический задел для открытой криптографии, первой реальной криптосистемой с открытым ключом считают алгоритм RSA (названный по имени авторов - Рон Ривест (Ronald Linn Rivest), Ади Шамир (Adi Shamir) и Леонард Адлеман (Leonard Adleman)).

Справедливости ради следует отметить, что в декабре 1997 г. была обнародована информация, согласно которой британский математик Клиффорд Кокс (Clifford Cocks), работавший в центре правительственной связи (GCHQ) Великобритании, описал систему, аналогичную RSA, в 1973 г., а несколькими месяцами позже в 1974 г. Малькольм Вильямсон изобрел математический алгоритм, аналогичный алгоритму Диффи – Хеллмана - Меркле.

Суть шифрования с открытым ключом заключается в том, что для шифрования данных используется один ключ, а для расшифрования другой (поэтому такие системы часто называют ассиметричными).

Основная предпосылка, которая привела к появлению шифрования с открытым ключом, заключалось в том, что отправитель сообщения (тот, кто зашифровывает сообщение), не обязательно должен быть способен его расшифровывать. Т.е. даже имея исходное сообщение, ключ, с помощью которого оно шифровалось, и зная алгоритм

шифрования, он не может расшифровать закрытое сообщение без знания ключа расшифрования.

Первый ключ, которым шифруется исходное сообщение, называется открытым и может быть опубликован для использования всеми пользователями системы. Расшифрование с помощью этого ключа невозможно. Второй ключ, с помощью которого дешифруется сообщение, называется секретным (закрытым) и должен быть известен только законному получателю закрытого сообщения.

Алгоритмы шифрования с открытым ключом используют так называемые необратимые или односторонние функции. Эти функции обладают следующим свойством: при заданном значении аргумента x относительно просто вычислить значение функции $f(x)$, однако, если известно значение функции $y = f(x)$, то нет простого пути для вычисления значения аргумента x. Однако не всякая необратимая функция годится для использования в реальных криптосистемах. Следует также отметить, что в самом определении необратимости функции присутствует неопределенность. Под необратимостью понимается не теоретическая необратимость, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства за обозримый интервал времени.

Поэтому чтобы гарантировать надежную защиту информации, к криптосистемам с открытым ключом предъявляются два важных и очевидных требования.

1. Преобразование исходного текста должно быть условно необратимым и исключать его восстановление на основе открытого ключа.
2. Определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне.

3.2. Алгоритм RSA

Диффи и Хеллман определили новый подход к шифрованию, что вызвало к жизни разработку алгоритмов шифрования, удовлетворяющих требованиям систем с открытым ключом. Одним из первых результатов был алгоритм, разработанный в 1977 году Роном Ривестом, Ади Шамиром и Леном Адлеманом и опубликованный в 1978 году. С тех пор алгоритм Rivest-Shamir-Adleman (RSA) широко применяется практически во всех приложениях, использующих криптографию с открытым ключом.

Алгоритм основан на использовании того факта, что задача факторизации является трудной, т.е. легко перемножить два числа, в то время как не существует полиномиального алгоритма нахождения простых сомножителей большого числа.

Первым этапом любого асимметричного алгоритма является создание получателем шифрограмм пары ключей: открытого и секретного. Для алгоритма RSA этап создания ключей состоит из следующих операций.

1. Выбираются два простых числа p и q.
2. Вычисляется произведение $n = p * q$.
3. Вычисляется функция Эйлера $\varphi(n)$.
4. Выбирается открытый ключ e, как произвольное число $(1 < e < \varphi(n))$, взаимно простое с результатом функции Эйлера.
5. Вычисляется секретный ключ d, как обратное число к e по модулю $\varphi(n)$, из соотношения $(d * e)\ mod\ \varphi(n) = 1$.
6. Публикуется открытый ключ (e, n) в специальном хранилище, где исключается возможность его подмены.

Шифрование и дешифрование имеют следующий вид для некоторого незашифрованного блока M и зашифрованного блока C.

$$C = M^e\ (mod\ n),$$
$$M = C^d\ (mod\ n).$$

3.3. Определение криптосистемы на основе неприводимых многочленов

Криптосистема основанная на неприводимых многочленах, аналогична алгоритму RSA, с тем лишь отличием, что вместо простых чисел используются неприводимые многочлены, и все вычисления производятся в факторкольце порожденном произведением этих неприводимых многочленов.

Первым этапом, так же как и в алгоритме RSA, является генерация открытого и секретного ключей. Создание ключей состоит из следующих операций.

1. Выбираются два неприводимых многочлена $g(x), h(x)$ над конечным полем F_q, где q – простое число.
2. Вычисляется произведение $f(x) = g(x) * h(x)$.
3. Вычисляется количество обратимых элементов факторкольца $\mathbb{F}_q[x]/(f)$, порожденного многочленом $f(x)$, $\varphi(f) = (q^{m_1} - 1) * (q^{m_2} - 1)$, где m_1 - степень многочлена $g(x)$, m_2 - степень многочлена $h(x)$.
4. Выбирается открытый ключ e, как произвольное число $(1 < e < \varphi(f))$, взаимно простое с количеством обратимых элементов.
5. Вычисляется секретный ключ d, как обратное число к e по модулю $\varphi(f)$, из соотношения $(d * e)\ mod\ \varphi(f) = 1$.

6. Публикуется открытый ключ $(e, f(x))$ в специальном хранилище, где исключается возможность его подмены.

Шифрование и дешифрование имеют следующий вид. Некоторым незашифрованным блокам M и зашифрованным блокам C, ставятся в соответствие многочлены принадлежащие факторкольцу $\mathbb{F}_p[x]/(f)$, $m(x)$ и $c(x)$ соответственно.

$$c(x) = m(x)^e \ (mod\ f(x)),$$
$$m(x) = c(x)^d \ (mod\ f(x)).$$

Пример. Рассмотрим конечное поле $\mathbb{F}_2$. Над данным конечным полем возьмем следующие неприводимые многочлены $g(x) = x^5 + x^2 + 1$ и $h(x) = x^6 + x^4 + x^2 + x + 1$. Вычислим произведение данных многочленов $f(x) = g(x) * h(x) = x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^3 + x + 1$. Количество обратимых элементов в факторкольце $\varphi(f) = (2^5 - 1) * (2^6 - 1) = 1953$. Выбираем открытый ключ $e = 101$, взаимно простой с $\varphi(f)$. Вычисляем секретный ключ d, из соотношения $(d * e)\ mod\ \varphi(f) = 1$, $d = 1895$.

Для шифрования было выбрано сообщение «Hello World!». Далее необходимо выбрать преобразование блоков исходного сообщения в многочлены принадлежащие факторкольцу $\mathbb{F}_2[x]/(f)$. Заметим, что так как степень многочлена $f(x)$ равна 11, следовательно, мы можем представить любое число до $2^{12} = 4096$ в виде некоторого многочлена $m(x)$. В свою очередь каждый символ исходного сообщения мы представим в виде числа до 2^{12}. В данном примере каждому символу будет соответствовать ASCII код соответствующего символа. В итоге каждому символу исходного сообщения будет соответствовать многочлен из факторкольца $\mathbb{F}_2[x]/(f)$.

$m_1(x) = "H" = 72 = x^6 + x^3$,

$m_2(x) = "e" = 101 = x^6 + x^5 + x^2 + 1$,

$m_3(x) = "l" = 108 = x^6 + x^5 + x^3 + x^2$,

$m_4(x) = "l" = 108 = x^6 + x^5 + x^3 + x^2$,

$m_5(x) = "o" = 111 = x^6 + x^5 + x^3 + x^2 + x + 1$,

$m_6(x) = " " = 32 = x^5$,

$m_7(x) = "W" = 87 = x^6 + x^4 + x^2 + x + 1$,

$m_8(x) = "o" = 111 = x^6 + x^5 + x^3 + x^2 + x + 1$,

$m_9(x) = "r" = 114 = x^6 + x^5 + x^4 + x$,

$m_{10}(x) = "l" = 108 = x^6 + x^5 + x^3 + x^2$,

$m_{11}(x) = "d" = 100 = x^6 + x^5 + x^2$,

$m_{12}(x) = "!" = 33 = x^5 + 1$.

После шифрования каждого многочлена $m_i(x)$ получим следующие многочлены $c_i(x) = m_i(x)^{101} \ (mod\ f(x))$.

$c_1(x) = x^9 + x^7 + x^5 + x^2 + x = 678,$

$c_2(x) = x^8 + x^5 + x^4 + x^3 + x + 1 = 315,$

$c_3(x) = x^8 + 1 = 257,$

$c_4(x) = x^8 + 1 = 257,$

$c_5(x) = x^9 + x^7 + x^3 + x^2 + x = 654,$

$c_6(x) = x^{10} + x^5 + x^3 + x^2 + x = 1070,$

$c_7(x) = x^8 + x^3 + x + 1 = 267,$

$c_8(x) = x^9 + x^7 + x^3 + x^2 + x = 654,$

$c_9(x) = x^9 + x^8 + x^5 + x^4 + x^2 = 820,$

$c_{10}(x) = x^8 + 1 = 257,$

$c_{11}(x) = x^9 + x^6 + x = 578,$

$c_{12}(x) = x^9 + x^8 + x^5 + x^3 + 1 = 809.$

При дешифровании каждого $c_i(x)$, получим многочлены $m_i(x) = c_i(x)^{1895} \ (mod\ f(x))$, соответствующие исходному сообщению.

4. АЛГОРИТМ ГЕНЕРАЦИИ НЕПРИВОДИМЫХ МНОГОЧЛЕНОВ

Задача построения неприводимых многочленов больших степеней над конечными полями является одной из сложных и важных проблем в компьютерной алгебре, теории кодирования, криптографии и теории конечных полей. Кроме того, последние достижения в этих областях пробудили еще больший интерес к предмету данных многочленов. Рассмотрим следующий рекурсивный алгоритм генерации неприводимых многочленов над конечными полями.

Теорема 1. [17] Пусть $f(x), g(x) \in F_q[x]$ взаимно простые многочлены и пусть $P(x) \in F_q[x]$ неприводимый многочлен степени n. Тогда произведение

$$F(x) = g^n(x) \cdot P(f(x)/\, g(x)) \tag{1}$$

является неприводимым многочленом над полем F_q тогда и только тогда, когда $f(x) - \alpha \cdot g(x)$ неприводим над F_{q^n} для некоторого корня $\alpha \in F_{q^n}$ многочлена $P(x)$.

Теорема 2. Пусть $x^p - b \cdot x + c$ и $x^p - b \cdot x + h$ – взаимно простые многочлены в кольце многочленов $F_q[x]$ и $P(x) = \sum_{i=0}^{n} c_i \cdot x^i$ – неприводимый многочлен степени $n \geq 2$ над конечным полем F_q. Также определим $b \in F_q^*$, $c, h \in F_q$, $(c, h) \neq (0, 0)$. Тогда

$$F(x) = (x^p - b \cdot x + h)^n \cdot P\left(\frac{x^p - b \cdot x + c}{x^p - b \cdot x + h}\right), \tag{2}$$

является неприводимым многочленом степени $p \cdot n$ над конечным полем F_q, если

$$N_{q/p}(b) = 1, \tag{3}$$

и

$$Tr_{q/p}\left(\frac{1}{A^p} \cdot \left(\frac{(c - h) \cdot P'(1)}{P(1)} + h \cdot n\right)\right) \neq 0, \tag{4}$$

где $A^{p-1} = b$ для некоторого $A \in F_{q^n}$ и $P'(1)$ является формальной производной многочлена $P(x)$ в точке 1.

В теореме 2 описан общий случай рекурсивного алгоритма генерации неприводимого многочлена. Так как подобрать трехчлены вида $x^p - b \cdot x + c$ и $x^p - b \cdot x + h$, так чтобы выполнялись условия (3,4) является нетривиальной задачей, рассмотрим частный случай этого алгоритма.

Теорема 3. Рассмотрим конечное поле F_p, где p – простое число. Пусть $x^p - x + \delta$ и $x^p - x + 1$ – взаимно простые многочлены в кольце многочленов $F_p[x]$ и $P(x)$ –

неприводимый многочлен степени $n \geq 2$ над конечным полем F_p. Также $\delta \in F_p$, $\delta + 1 \neq 0$. Определим,

$$F_0(x) = P(x). \tag{5}$$

$$F_k(x) = (x^p - x + 1)^{n_{k-1}} \cdot F_{k-1}\left(\frac{x^p - x + \delta}{x^p - x + 1}\right), k \geq 1, \tag{6}$$

где $n_k = n \cdot p^k$ определена как степень многочлена $F_k(x)$. Предположим, что $(\delta - 1) \cdot P'(1) + n \cdot P(1) \neq 0$ и $(\delta - 1) \cdot P'(\delta) - n \cdot P(\delta) \neq 0$. Тогда $F_k(x)$ является неприводимым многочленом степени $n \cdot p^k$ над полем F_p для каждого $k \geq 1$.

Пример. Возьмем неприводимый многочлен $P(x) = x^4 + x^2 + 2x + 2$, над конечным полем F_5. Пусть $\delta = 2$, проверим будут ли выполняться условия для генерации неприводимых многочленов.

1. Многочлены $x^5 - x + 2$ и $x^5 - x + 1$ взаимно простые над конечным полем F_5.
2. $P'(x) = 4x^3 + 2x + 2, P'(1) = 3, P(1) = 1.$
   $(\delta - 1) \cdot P'(1) + n \cdot P(1) = 2.$
3. $P'(\delta) = 3, P(\delta) = 1.$
   $(\delta - 1) \cdot P'(\delta) - n \cdot P(\delta) = 4.$

Так как все условия для генерации многочленов выполняются можно приступить к следующему шагу алгоритма.

Вычислим значение многочлена

$$F_0\left(\frac{x^5 - x + 2}{x^5 - x + 1}\right) = P\left(\frac{x^5 - x + 2}{x^5 - x + 1}\right)$$

$$= \frac{(x^{20} + 2x^{16} + x^{15} + x^{12} + 4x^{11} + 2x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + 3x^3 + 2x^2 + 4x + 1)}{(x^{20} + x^{16} + 4x^{15} + x^{12} + 3x^{11} + x^{10} + x^8 + 2x^7 + 3x^6 + 4x^5 + x^4 + x^3 + x^2 + x + 1)},$$

и значение многочлена

$$(x^5 - x + 1)^4 = x^{20} + x^{16} + 4x^{15} + x^{12} + 3x^{11} + x^{10} + x^8 + 2x^7 +$$
$$3x^6 + 4x^5 + x^4 + x^3 + x^2 + x + 1.$$

Результатом произведения этих двух многочленов будет следующий многочлен $F_1(x)$.

$$F_1(x) = x^{20} + x^{16} + 2x^{15} + x^{12} + 4x^{11} + 2x^{10} + x^8 + x^7 + x^6 + x^5 +$$
$$x^4 + 3x^3 + 2x^2 + 4x + 1.$$

Данный многочлен является неприводимым, проверка производилась алгоритмом Берлекэмпа.

ЗАКЛЮЧЕНИЕ

В рамках работы были рассмотрены неприводимые многочлены над конечными полями, строение факторкольца кольца многочленов, алгоритмы факторизации многочленов над конечными полями.

Рассмотрены существующие криптосистемы с открытым ключом (алгоритм RSA), и на их основе разработана криптосистема, основанная на факторкольце кольца многочленов. Также рассмотрен рекурсивный алгоритм генерации неприводимых многочленов над конечными полями и приведен пример работы данного алгоритма.

Реализованы в системе компьютерной алгебры GAP алгоритм факторизации многочленов над конечными полями (алгоритм Берлекэмпа), криптосистема на основе факторкольца кольца многочленов и рекурсивный алгоритм генерации неприводимых многочленов над конечным полем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Баранский В.А. Введение в общую алгебру и ее приложения. / В.А. Баранский. Екатеринбург, 1998 г. 169 с.

2. Болотов А.А. Элементарное введение в эллиптическую криптографию. Алгебраические и алгоритмические основы / А.А. Болотов, С.Б. Гашков, А.Б. Фролов. М.: КомКнига, 2012. 328 с.

3. Болотов А.А. Элементарное введение в эллиптическую криптографию. Протоколы криптографии на эллиптических кривых / А.А. Болотов, С.Б. Гашков, А.Б. Фролов. М.: КомКнига. 2006. 280 с.

4. Берлекэмп Э. Алгебраическая теория кодирования / Э. Берлекэмп – М.: Мир, 1971. – 478 с.

5. Вейль А. Основы теории чисел / А.Вейль. М.: Мир. 1972. 408 с.

6. Виноградов И.М. Основы теории чисел / И.М. Виноградов. Москва-Ижевск: НИЦ «Регулярная и хаотическая динамика». 2003. 176 с.

7. Гараков Г.А. Таблицы неприводимых полиномов над GF(p) ($p \leq 11$) / Г.А. Гараков // Математические вопросы кибернетики и вычислительной техники. Труды ВЦ АН Арм.ССР и Ереван.гос.унив. №6. 1970. С. 112-142.

8. Геут Кр.Л. О генерации неприводимых многочленов / Кр.Л. Геут, С.С. Титов // Математическое моделирование системных взаимосвязей в прикладных исследованиях : сб. науч. тр. Екатеринбург : Изд-во УрГУПС, 2013. Вып. 13(196). С. 12-18.

9. Геут Кр.Л. Построение неприводимых многочленов простого порядка / Кр.Л. Геут, С.С. Титов // Безопасность информационного пространства : материалы XII Всерос. науч.-практ. конф. Екатеринбург: Изд-во Урал. ун-та, 2014. С. 219-225.

10. Иванов М.А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М.А. Иванов, И.В. Чугунков. М.:КУДИЗ-ОБРАЗ, 2003. 240 с.

11. Крайнюков Н.И. Основы системы компьютерной алгебры GAP / Н.И. Крайнюков, Б.Ф. Мельников: Тольятти: ТГУ, 2012. – 26 с.

12. Кузьмин Е.Н. О неприводимых многочленах над конечным полем / Е.Н. Кузьмин // Алгебра и логика. 1994. Т. 33, №4. С. 387-414.

13. Лидл Р. Конечные поля / Р. Лидл, Г. Нидеррайтер: В 2 т. – Т. 1. – М.: Мир, 1988. – 820 с.

14. Рябко, Б.Я. Основы современной криптографии для специалистов в информационных технологиях / Б.Я. Рябко, А.Н. Фионов. – М.: Научный мир, 2004. – 173 с.

15. Харин Ю.С. Математические и компьютерные основы криптологии. / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. Минск: Новое знание. 2003. 382 с.

16. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке Си / Б. Шнайер – М.: Триумф, 2002. – 816 с.

17. Abrahamyan S. Recursive constructions of irreducible polynomials over finite fields / S. Abrahamyan, M. Alizadeh, M. K. Kyureghyan // Finite Fields and Their Applications 18 (2012), 738-745pp.

18. Koblitz N. Algebraic aspects of Cryptography. / N. Koblitz // Springer. 2004. 109 pp.

ПРИЛОЖЕНИЕ 1

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра дифференциальных и стохастических уравнений

НЕПРИВОДИМЫЕ МНОГОЧЛЕНЫ НАД КОНЕЧНЫМИ ПОЛЯМИ И ИХ ПРИМЕНЕНИЕ В КРИПТОГРАФИИ

ТЕКСТ ПРОГРАММЫ
ЮУрГУ – 09.04.04.2016.129-129.ВКР

Руководитель, доктор физ.-мат. наук, доцент
_____ / Н.Д. Зюляркина /
«07» июня 2016 г.

Автор, студент группы ММиКН-293
_____ / А.В. Корсун /
«02» июня 2016 г.

Нормоконтролер, канд. физ.-мат. наук, доцент
_____ / М.А. Сагадеева /
«___» _____ 2016 г.

Челябинск 2016

УДК 512.624.2

Корсун А.В.

Неприводимые многочлены над конечными полями и их применение в криптографии. /А.В. Корсун. – Челябинск, 2016. - 14 с. .

В данном приложении приводится исходный текст программ, реализованных в системе компьютерной алгебры GAP. Реализованы алгоритм факторизации многочленов над конечным полем (алгоритм Берлекэмпа), рекурсивный алгоритм генерации неприводимых многочленов и криптосистема, основанная на факторкольце кольца многочленов.

ОГЛАВЛЕНИЕ

П1.1 ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ ФАКТОРИЗАЦИИ МНОГОЧЛЕНОВ

П1.1.1 Исходный текст файла «main.g»

```
p := 13;
ZN := Integers mod p;
x := Indeterminate(ZN, "x");
arr := [0, 0, 1, 3, 1];
n := Length(arr) - 1;
fl_func := Filename( DirectoriesLibrary("bin"), "functions.g" );;
Read(fl_func);
q := get_Q(arr, n, p);
rr := calc(q, n, p);
f := build(arr);
buf := [];
for i in [1..Length(rr)] do
     Add(buf, build(rr[i]));
od;
res := solve(f, buf, Length(rr));
Print(res);
```

П1.1.2 Исходный текст файла «functions.g»

```
get_obr := function(numb, p)
     local g, res;
     g := Gcdex(numb, p);
     res := ((g.coeff1 mod p) + p) mod p;
     return res;
end;

normalize := function(numb, p)
     numb := ((numb mod p) + p) mod p;
     return numb;
end;

build := function(arr)
     local res, n, i;
     res := 0;
     n := Length(arr);
     for i in [1..n] do
          res := res + (arr[i] * x^(i-1));
     od;
     return res;
end;

solve := function (f, arr, n)
     local res, t, m, i, j, k, h, buf;
     buf := build([1]);
     t := 1;
     m := 1;
     res := [];
     Add(res, f);
     for i in [1..n] do
          if t > n then
               break;
          fi;
```

```
            for j in [2..n] do
                if t > n then
                    break;
                fi;
                for k in [0..p-1] do
                    if t > n then
                        break;
                    fi;
                    h := Gcd(res[i], arr[j] - k);
                    if h <> buf then
                        if h = res[i] then
                            break;
                        else
                            t := t + 1;
                            Add(res, h);
                            res[i] := res[i] / h;
                        fi;
                    fi;
                od;
            od;
        od;
        return res;
end;


calc := function(q, n, p)
        local c, prev, res, buf, ind, obr, i, j, k, abc;
        c := [];
        prev := [];
        res := [];
        for i in [1..n] do
            Add(c, -1);
            Add(prev, []);
        od;

        for k in [1..n] do
            ind := -1;
            for j in [1..n] do
                if (q[k][j] <> 0) and (c[j] < 0) then
                    ind := j;
                    break;
                fi;
            od;
            if ind <> -1 then
                obr := get_obr(q[k][ind], p);
                for i in [1..n] do
                    q[i][ind] := q[i][ind] * (-1) * obr;
                    q[i][ind] := normalize(q[i][ind], p);
                od;

                for i in [1..n] do
                    prev[i] := ShallowCopy(q[i]);
                od;

                for j in [1..n] do
                    if j <> ind then
                        for i in [1..n] do
```

```
                                        q[i][j] := prev[i][j] + (prev[k][j] *
prev[i][ind]);
                                        q[i][j] := normalize(q[i][j], p);
                                   od;
                              fi;
                         od;
                         c[ind] := k;
                    else
                         buf := [];
                         for i in [1..n] do
                              Add(buf, 0);
                         od;
                         for i in [1..n] do
                              j := c[i];
                              if j >= 0 then
                                   buf[j] := q[k][i];
                              fi;
                         od;
                         buf[k] := 1;
                         Add(res, buf);
                    fi;
               od;
               return res;
          end;

          get_Q := function(arr, n, p)
               local q, cur, prev, count, ptr, a, b, c, j, i;
               q := [];
               cur := [];
               for i in [1..n] do
                    Add(q, []);
                    Add(cur, 0);
               od;
               cur[1] := 1;
               prev := ShallowCopy(cur);

               q[1] := ShallowCopy(cur);
               count := p * (n-1);
               ptr := 2;

               for i in [1..count] do
                    for j in [1..n] do
                         a := 0;
                         if j <> 1 then
                              a := prev[j-1];
                         fi;
                         b := prev[n];
                         c := arr[j];
                         cur[j] := a - b * c;
                         cur[j] := normalize(cur[j], p);
                    od;
                    if (i mod p) = 0 then
                         q[ptr] := ShallowCopy(cur);
                         ptr := ptr + 1;
                    fi;
                    prev := ShallowCopy(cur);
               od;
```

```
        for i in [1..n] do
            q[i][i] := q[i][i] - 1;
            q[i][i] := normalize(q[i][i], p);
        od;
        return q;
end;
```

П1.2 ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ РЕАЛИЗУЮЩЕЙ КРИПТОСИСТЕМУ НА ОСНОВЕ ФАКТОРКОЛЬЦА КОЛЬЦА МНОГОЧЛЕНОВ

П1.2.1 Исходный текст файла «main.g»

```
p := 2;
ZN := Integers mod p;
x := Indeterminate(ZN, "x");
one := x / x;
zero := one - one;
px := x^5 + x^2 + 1;
qx := x^6 + x^4 + x^2 + x + 1;
fx := px * qx;
n := DegreeIndeterminate(fx, x);

fl_polynomial := Filename( DirectoriesLibrary("bin"),
     "polynomial.g" );;
Read(fl_polynomial);

polynomials := getAllPolynomials(n);
countInvertiblePolynomials :=
getCountInvertiblePolynomials(polynomials);

fl_message := Filename( DirectoriesLibrary("bin"), "message.g" );;
Read(fl_message);

e := 101;
c := e ^ (-1) mod countInvertiblePolynomials;

message := "Hello World!!!";
arrayPolynomials := messageToPolynomials(message);

encryptArrayPolynomials := encrypt(arrayPolynomials, e, fx);
decryptArrayPolynomials := decrypt(encryptArrayPolynomials, c, fx);

decryptMessage := polynomialsToMessage(decryptArrayPolynomials);
Print(decryptMessage);
```

П1.2.2 Исходный текст файла «polynomial.g»

```
binpow := function(fx, n, gx)
     local res, m;
     res := x / x;
     while n <> 0 do
          m := n mod 2;
          if m <> 0 then
               res := (res * fx) mod gx;
          fi;
          fx := (fx * fx) mod gx;
          n := (n - m) / 2;
     od;
     return res;
end;

recursionGetPolynomials := function (numb, st, res, buf)
```

```
        local cur, i;
        if numb = st + 1 then
              cur := 0;
              for i in [1..st] do
                    cur := cur + (buf[i] * (x ^ (i-1)));
              od;
              if cur <> zero then
                    Add(res, cur);
              fi;
              return 0;
        fi;
        for i in [0..p-1] do
              buf[numb] := i;
              recursionGetPolynomials(numb + 1, st, res, buf);
        od;
  end;

  getAllInvertiblePolynomials := function(array)
        local res, len, i, j, aa, bb, cc;
        res := [];
        len := Length(array);
        for i in [1..len] do
              aa := array[i];
              for j in [i..len] do
                    bb := array[j];
                    cc := (aa * bb) mod fx;
                    if (cc = one) then
                          if (aa = bb) then
                                Add(res, aa);
                          else
                                Add(res, aa);
                                Add(res, bb);
                          fi;
                    fi;
              od;
        od;
        return res;
  end;


  getAllPolynomials := function(st)
        local res, buf;
        res := [];
        buf := [];
        recursionGetPolynomials(1, st, res, buf);
        return res;
  end;


  getCountInvertiblePolynomials := function(array)
        local res, invertiblePolynomials;
        invertiblePolynomials := getAllInvertiblePolynomials(array);
        res := Length(invertiblePolynomials);
        return res;
  end;
```

П1.2.3 Исходный текст файла «message.g»

```
stringToInt := function(str)
      local res, len, i;
      res := [];
      len := Length(str);
      for i in [1..len] do
            Add(res, IntChar(str[i]));
      od;
      return res;
end;

intToString := function(array)
      local res, len, i;
      res := [];
      len := Length(array);
      for i in [1..len] do
            Add(res, CharInt(array[i]));
      od;
      return res;
end;

intToByteArray := function(numb)
      local res, ost;
      res := [];
      while numb <> 0 do
            ost := numb mod 2;
            if ost <> 0 then
                  Add(res, 1);
            else
                  Add(res, 0);
            fi;
            numb := (numb - ost) / 2;
      od;
      return res;
end;

byteArrayToInt := function(array)
      local res, cur, len, i;
      res := 0;
      cur := 1;
      len := Length(array);
      for i in [1..len] do
            res := res + (array[i] * cur);
            cur := cur * 2;
      od;
      return res;
end;

byteArrayToPolynomial := function(byteArray)
      local polynomial, len, number, i;
      polynomial := 0;
      len := Length(byteArray);
      for i in [1..len] do
            number := byteArray[i];
            polynomial := polynomial + (number * x^(i-1));
      od;
      return polynomial;
```

```
end;

polynomialToByteArray := function(polynomial)
      local byteArray, coefficients, len, i;
      byteArray := [];
      coefficients := PolynomialCoefficientsOfPolynomial(polynomial,x);
      len := Length(coefficients);
      for i in [1..len] do
            if coefficients[i] = one then
                  Add(byteArray, 1);
            else
                  Add(byteArray, 0);
            fi;
      od;
      return byteArray;
end;

messageToPolynomials := function(message)
      local arrayPolynomials, messageInt, len, number, byteArray,
polynomial, i;
      arrayPolynomials := [];
      messageInt := stringToInt(message);
      len := Length(messageInt);
      for i in [1..len] do
            number := messageInt[i];
            byteArray := intToByteArray(number);
            polynomial := byteArrayToPolynomial(byteArray);
            Add(arrayPolynomials, polynomial);
      od;
      return arrayPolynomials;
end;


polynomialsToMessage := function(arrayPolynomials)
      local message, messageInt, len, number, byteArray, polynomial, i;
      len := Length(arrayPolynomials);
      messageInt := [];
      for i in [1..len] do
            polynomial := arrayPolynomials[i];
            byteArray := polynomialToByteArray(polynomial);
            number := byteArrayToInt(byteArray);
            Add(messageInt, number);
      od;
      message := intToString(messageInt);
      return message;
end;

encrypt := function(array, e, fx)
      local res, len, i, buf;
      res := [];
      len := Length(array);
      for i in [1..len] do
            buf := binpow(array[i], e, fx);
            Add(res, buf);
      od;
      return res;
end;
```

```
decrypt := function(array, c, fx)
      local res, len, i, buf;
      res := [];
      len := Length(array);
      for i in [1..len] do
            buf := binpow(array[i], c, fx);
            Add(res, buf);
      od;
      return res;
end;
```

П1.3 ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ ФАКТОРИЗАЦИИ МНОГОЧЛЕНОВ

П1.3.1 Исходный текст файла «check.g»

```
p := 5;
ZN := Integers mod p;
x := Indeterminate(ZN, "x");
one := x / x;
zero := one - one;

px := x^4 + x^2 + 2*x + 2;
qx := Derivative(px);

n := Degree(px);
a := Value(qx, 1);
b := Value(px, 1);
flag := 0;

for i in [1..n-1] do
     delta := i;

     c := Value(qx, delta);
     d := Value(px, delta);

     e := (delta - 1);

     r1 := (e * a) + (n * b);
     r2 := (e * c) - (n * d);

     f1 := x^p - x + delta;
     f2 := x^p - x + 1;

     if (r1 <> 0) and (r2 <> 0) and (Gcd(f1, f2) = one) then
          Print(delta);
          flag := 1;
          break;
     fi;
od;

if flag = 0 then
     Print("Для заданного неприводимого многочлена не существует ни
     одного подходящего delta!");
fi;
```

П1.3.2 Исходный текст файла «generate.g»

```
p := 5;
ZN := Integers mod p;
x := Indeterminate(ZN, "x");
one := x / x;
zero := one - one;

PrintPolynomial := function(pol)
     local flag, buf, coeff, degree, buf2;
     flag := 0;
     buf := pol;
```

```
        while buf <> zero do
              coeff := IntFFE(LeadingCoefficient(buf));
              degree := Degree(buf);

              if flag = 0 then
                    flag := 1;
              else
                    Print("+");
              fi;

              if coeff <> 1 then
                    Print(coeff, "*");
              fi;

              Print("x^", degree);

              buf2 := coeff * (x ^ degree);
              buf := buf - buf2;
        od;
  end;

  countStep := 3;
  px := x^4 + x^2 + 2*x + 2;
  n := Degree(px);

  fx := x^5 - x + 2;
  gx := x^5 - x + 1;
  hx := fx / gx;

  for i in [1..countStep] do
        buf := px;
        ch := zero;
        while buf <> zero do
              coeff := LeadingCoefficient(buf);
              degree := Degree(buf);

              add := (hx ^ degree) * coeff;
              ch := ch + add;

              buf2 := coeff * (x ^ degree);
              buf := buf - buf2;
        od;

        h2 := gx ^ (n * (p ^ (i-1)));
        px := h2 * ch;
        PrintPolynomial(px);
        Print("\n\n");
  od;
```