Содержание

1 Введение

Задача построения базиса Грёбнера является важной задачей вычислительной алгебры, теории чисел и алгебраической геометрии. В 1965 году Бухбергер в работе [Buch] предложил конструктивный алгоритм построения базиса Грёбнера, однако время его работы в общем случае довольно велико.
В работах [Faug1], [Faug2] Ж.-Ш.Фожер предложил алгоритмы $F4$ и $F5$, вычисляющие базис Грёбнера идеала кольца многочленов от нескольких переменных более быстрым способом. В 2005 году в работе [Steg].Штегерс более подробно изучил алгоритм $F5$, а также сделал попытку (неудачную) соединения алгоритмов $F4$ и $F5$ для достижения большей эффективности.
В работах Фожера были найдены недочёты, и впоследствии в работах [Gal], [Ger], [EdPer]были сформулированы и доказаны различные утверждения об алгоритме $F5$.
В настоящей работе дан обзор алгоритма Бухбергера, сделана попытка структурировать описание алгоритма $F5$ и доказательство его останавливаемости аналогично статье [Gal], а также исследованы границы его применения к задаче дискретного логарифмирования на эллиптической кривой.

2 Алгоритм Бухбергера

Пусть K – поле, и I – идеал кольца многочленов $K[x_1, \ldots, x_n]$. Пусть также на множестве мономов T задан полный порядок $<$, удовлетворяющий следующим свойствам:

1. $\forall t \in T\ (t \neq 1)\ 1 < t$;
2. $\forall t, t_1, t_2 \in T, t_1 < t_2 \Rightarrow tt_1 < tt_2$.

Для ненулевого многочлена $f = \sum_{t \in T} a_t t$ обозначим за $l_t(f)$ или $\mathrm{HT}(f)$ наибольший элемент t, такой что $a_t \neq 0$. Такой элемент называется главным термом f, а соответствующий ему коэффициент – главным коэффициентом $l_c(f)$ или $\mathrm{HC}(f)$. Для нулевого многочлена положим главный терм и главный коэффициент равными нулю.

Определение 1. *Базисом Грёбнера идеала I относительно порядка $<$ называется множество G, такое что для любого $f \in I$ найдутся $s \in T$ и $g \in G$, что $l_t(f) = s\, l_t(g)$.*

Также часто встречающимся является понятие редуцированного базиса Грёбнера. Пусть $H \subset K[x_1, \ldots, x_n]$, а f – ненулевой многочлен.

Определение 2. *f редуцируем по модулю H, если в f есть моном t, такой что $t = s\, l_t(g)$ для некоторых $s \in T, g \in H$.*

Определение 3. *Редуцированный базисом Грёбнера идеала I называется такой базис Грёбнера G, что для любого $g \in G$ многочлен g нельзя редуцировать по модулю $G \setminus \{g\}$ и $l_c(g) = 1$.*

Для введения алгоритма Бухбергера необходимо также определить понятие s-многочлена.

Определение 4. *s-многочленом f и g называется многочлен вида:*

$$spol(f, g) = \mathrm{lcm}(l_t(f), l_t(g))\left(\frac{f}{l_t(f)} - \frac{g}{l_t(g)}\right)$$

Опишем алгоритм Бухбергера:
Пусть F – множество многочленов, порождающее идеал I. Наша задача – построить множество G, являющееся базисом Грёбнера для I. Алгоритм состоит из следующих шагов:

1. Возьмем $G = F$.
2. Для всех g_i, g_j из G найдем многочлен $spol(g_i, g_j)$.
3. Если $spol(g_i, g_j)$ редуцируем по модулю G, то редуцируем его до тех пор, пока результат не перестанет быть редуцируемым.

4. Если результат шага 3 ненулевой, то добавляем его в G.

5. Повторяем шаги 2-4, пока все пары (включая пары с добавленными на шаге 3 многочленами) не будут рассмотрены.

6. Получившееся множество G будет базисом Грёбнера идеала I.

Остановка алгоритма гарантируется тем, что в процессе работы размер идеала, порожденного главными членами элементов G, увеличивается, а согласно лемме Диксона любая неубывающая цепочка идеалов начиная с некоторого момента становится константной.

Рассматривая алгоритм, можно заметить, что если на шаге 3 s-многочлен редуцируется к нулю, то в множество не добавляется ничего нового, при этом затраты на операцию редукции достаточно велики. Таким образом, основным направлением поиска улучшения алгоритма Бухбергера является возможность заранее сказать, редуцируется ли некий многочлен к нулю или нет. Ж.-Ш. Фожер в 1999 и 2002 предложил алгоритмы $F4$ и $F5$, позволяющие избежать редукций к нулю благодаря проверке, удовлетворяет ли многочлен некоторым критериям. В настоящей работе будет рассмотрен алгоритм $F5$.

3 Отмеченные многочлены

Перед тем, как описать алгоритм $F5$, необходимо ввести несколько новых определений.
Пусть $I = (f_1, \ldots, f_n), I_i = (f_i, \ldots, f_n), I_1 = I$.

Определение 5. *Индексом многочлена f называется максимальное число i, такое что I_i содержит f: $i(f) = \max\{i \mid f \in I_i\}$.*

Сигнатурой многочлена f называется минимальный из главных термов многочленов, обладающих следующим свойством: f сравнимо по модулю $I_{i(f)+1}$ с произведением p на $f_{i(f)}$: $s(f) = \min\{\mathrm{HT}(p) \mid f \equiv pf_{i(f)} (\mathrm{mod} I_{i(f)+1})\}$.

Меткой многочлена f называется пара из индекса и сигнатуры.

Отмеченным многочленом f называется пара из метки и многочлена. Многочлен при этом обозначается $poly(f)$, а метка – $M(f)$.

Далее необходимо ввести порядок на метках и операцию умножения терма на метку.
Порядок: $M(f) < M(g)$, если $i(f) > i(g)$ или $i(f) = i(g), s(f) < s(g)$.
Умножение терма на метку: $t \cdot M(f) = (i(f), t \cdot s(f))$.
Будем говорить, что $f = \mathfrak{o}_G(h)$, если существует представление $f = \sum_{g \in G} \lambda_g g$, такое что для любого g выполняются два неравенства: $\mathrm{HT}(\lambda_g g) < \mathrm{HT}(h)$ и $M(\lambda_g g) < M(h)$.
Также необходимо сформулировать понятие критической пары:

Определение 6. *Критической парой относительно множества G называется такая пара отмеченных многочленов f и g, что $spol(f, g) = \mathfrak{o}_G(u_{f,g} f)$, где $u_{f,g} = \dfrac{\mathrm{lcm}(\mathrm{HT}(f), \mathrm{HT}(g))}{\mathrm{HT}(f)}$.*

4 Алгоритм F5

В этом разделе будут описаны процедуры, составляющие алгоритм F5.

1. Rewritten?

   Данная процедура проверяет выполнение так называемого "критерия перезаписи" для произведения монома u на отмеченный многочлен f_k. Соответственно, результатом работы процедуры является булево значение True, если $u \cdot f_k$ отбрасывается критерием перезаписи, и False в противоположном случае.
   Критерий перезаписи проверяется для всех отмеченных многочленов из G с той же сигнатурой, что и у многочлена f_k. Для всех таких многочленов проводится следующая проверка: если в G найдется многочлен $f_j \neq f_k$, такой что $i(f_j) = i(f_k)$ и $us(f_k) \vdots s(f_j)$, то произведение отбрасывается критерием перезаписи. Если же ни для одного из многочленов из G оба эти условия одновременно не выполняются, то произведение не отбрасывается критерием перезаписи.

2. CritPair

   Эта процедура создаёт критическую пару отмеченных многочленов (в случае соответствия входных данных некоторым заданным условиям) или возвращает пустое множество (в случае несоответствия входных данных). На вход процедуре даются два отмеченных многочлена f_k, f_l из G, число i – индекс сигнатур, для которых будет проверяться редуцированность и множество L, относительно которого проверяется редукция.

   Сначала находим наименьшее общее кратное старших мономов многочленов f_k и f_l. Обозначим это кратное за t. Далее создаём мономы u_1, u_2 из формулы для s-многочлена:

   $$u_1 = \frac{1}{\mathrm{HC}(f_k)} \frac{t}{\mathrm{HM}(f_k)}, u_2 = \frac{1}{\mathrm{HC}(f_l)} \frac{t}{\mathrm{HM}(f_l)},$$

   а за t_1, t_2 обозначим сигнатуры многочленов f_k и f_l. Дальше, если $u_1 M(f_k) < u_2 M(f_l)$ (где $<$ – порядок на метках), то поменяем местами k и l и снова запустим ту же процедуру. Далее, если $\varphi(u_1 t_1) \neq u_1 t_1$, где φ – оператор редукции по множеству L, и индекс $f_k = i$, то процедура возвращает пустое множество. Также процедура возвращает пустое множество если $\varphi(u_2 t_2) \neq u_2 t_2$ и при этом индекс $f_l = i$. Если же все эти условия не выполнены, то процедура возвращает критическую пару $(u_1 \cdot f_k, u_2 \cdot f_l)$.

3. Spol

   Эта процедура по множеству s-пар фиксированной степени составляет s-многочлены и добавляет их в множество G. Обозначим исходное множество

за P. Для достижения цели для всех элементов P проведем следующую процедуру:

Обозначим рассматриваемый на очередном шаге элемент массива как пару $(u \cdot f, v \cdot g)$ (то, что каждый элемент P имеет такой вид, гарантируется процедурой CritPair). Если обе пары $(u, f), (v, g)$ не отбрасываются критерием перезаписи, то добавляем в G многочлен $(uM(f), u\text{poly}(f) - v\text{poly}(g))$.

4. IsReducible

Процедура проверки, "просеивающая" множество потенциальных редукторов G_{red} в поисках сигнатурной редукции для многочлена f_k при данном операторе редукции φ по множеству G. Результатом проверки является пустое или одноэлементное множество потенциальных редукторов.

Пусть r_j – очередной элемент G_{red} и до этого процедура не завершилась. Тогда IsReducible вернёт множество, содержащее только r_j в случае выполнения следующих 4 условий:

a. В дроби $\dfrac{\text{HT}(f_k)}{\text{HT}(r_j)}$ числитель делится на знаменатель, то есть эта дробь (обозначим её за u) – корректный моном;

b. $\varphi(ut_j) = ut_j$, где t_j – моном сигнатуры многочлена r_j;

c. Многочлен (u, r_j) не отбрасывается критерием перезаписи;

d. $u\text{S}(r_j) \neq \text{S}(f_k)$.

Если же в G_{red} не нашлось элемента, который бы удовлетворял всем этим 4 условиям, то процедура возвращает пустое множество.

5. TopReduction

Эта процедура выполняет топ-редукцию – сигнатурную редукцию многочлена f_k. При этом редукторами являются многочлены из множества G_{red}, а оператор редукции обозначен за ϕ. Результатом работы процедуры, помимо перезаписанного многочлена f_k, является пара множеств (ToDo, Done), каждое из которых содержит некоторые многочлены из G.

Для начала необходимо убедиться, что полиномиальная часть f_k ненулевая. В противном случае многочлен не перезаписывается, а множества ToDo и Done остаются пустыми. В дальнейшем предполагаем, что эта "нулевая"проверка пройдена.

На место многочлена f_k запишем многочлен $(M(f_k), \frac{\text{poly}(f_k)}{\text{HC}(f_k)})$ (отнормируем f_k), после чего проверим возможность дальнейшей редукции с помощью

уже описанной нами процедуры IsReducible. Если она невозможна, то множество ToDo оставим пустым, а в множество Done запишем единственный элемент – f_k, и на этом прекратим выполнение. Иначе процедура IsReducible вернула одноэлементное множество. Обозначим этот элемент за r.

Обозначим дробь $\frac{\mathrm{HM}(f_k)}{\mathrm{HM}(r))}$ за u. Если $u\mathrm{S}(r) < \mathrm{S}(f_k)$, то снова перезапишем f_k, на этот раз многочленом $(M(f_k), \mathrm{poly}(f_k) - u\mathrm{poly}(r)))$, массив Done оставим пустым, а в массив ToDo запишем единственный элемент – f_k, и на этом прекратим выполнение.

Если же это условие не выполняется, то необходимо добавлять новый многочлен в G, а именно многочлен $(uM(r), \mathrm{poly}(f_k) - u\mathrm{poly}(r))$. После этого, массив Done оставим пустым, а в массив ToDo запишем два элемента – f_k и f_n (то есть только что добавленный в G многочлен), и на этом прекратим выполнение процедуры.

6. Reduction

Эта процедура отвечает за все редукции внутри алгоритма. В процессе её работы многочлены с позициями из множества ToDo редуцируются по многочленам из множества G. В результате получается множество Done, содержащее редуцированные многочлены из ToDo по многочленам из G.

Процедура итерируется по многочленам из ToDo, упорядоченным по возрастанию сигнатур. Обозначим многочлен на очередном шаге за h. Далее делаем следующее: выкинув из множества ToDo многочлен h, редуцируем его – вместо него запишем многочлен $(M(h), \varphi(\mathrm{poly}(h)))$. После этого попытаемся провести процедуру топ-редукции, использовав в качестве множества потенциальных редукторов не только всё множество G, но и уже построенные элементы множества Done. Результатом работы будут множества Done1 и ToDo1, которые мы присоединим к соответствующим множествам Done и ToDo. Эту процедуру будем повторять до тех пор, пока множество ToDo не опустеет.

Заметим, что если многочлен используется в создании s-пары в качестве старшей по сигнатуре части, то в дальнейшем он не редуцируется (поскольку будет лежать либо в G_i, либо в Done.

7. AlgorithmF5

Данная процедура является основной частью алгоритма F5: она запускается в общем цикле для всё большего исходного множества (как именно, будет описано в следующей процедуре). Начальными данными для этой процедуры является отмеченный многочлен с мономом сигнатуры, равном 1, индексом сигнатуры равным некоторому числу i и полиномиальной частью f_i;

множество многочленов из G, которые образуют ранее вычисленный базис Грёбнера – G_{i+1}. Процедура общего цикла обеспечивает, что многочлены на этих места имеют индексы сигнатур, большие i.

Результатом работы этой процедуры на очередном шаге будет множество многочленов из G, которые образуют базис Грёбнера для расширенного на этом шаге идеала, порождённого многочленами $f_i, ..., f_m$.

Перед основной работой процедуры происходят три действия:

(a) Множество G_i получается объединением множества G_{i+1} и многочлена f_i;

(b) За φ обозначается оператор, производящий полную редукцию по множеству из полиномиальных частей отмеченных многочленов из G_{i+1};

(c) За P обозначается отсортированное по возрастанию сокращаемого монома множество, получившееся в результате применения процедуры CritPair к многочленам f_i и f_k при индексе сигнатур равном i и с оператором редукции φ, где f_k пробегает всё множество G_{i+1}.

Теперь начинается основная часть процедуры, которая работает итеративно по множеству P до его опустошения. По циклу происходят следующие действия: сначала в переменную d, которая будет содержать минимальную степень оставшихся s-пар, записывается степень сокращаемого монома первого элемента из множества P, а в множество P_d кладутся все многочлены из P, степень сокращаемых мономов которых равна d, после чего эти элементы удаляются из множества P. Далее к P_d применяется процедура Spol, что приводит к созданию множества F_d, в котором находятся s-многочлены, созданные из пар многочленов из P_d. Это множество F_d далее участвует в процедуре Reduction в качестве множества ToDo, а множеством, по которому происходят редукции, является множество G_i.

В результате этого действия создаётся множество R_d редуцированных многочленов. Для каждого элемента R_d (обозначим такой элемент на очередном шаге за r), во-первых, выполняется процедура CritPair, которая создаёт новые критические пары многочленов r и p, где p пробегает всё множество G_i, и присоединяет их к множеству P, во-вторых, добавляет все элементы r во множество G_i. После этих действий P снова упорядочивается по возрастанию сокращаемого монома и процедура повторяется итеративно.

8. IncrementalF5

Эта процедура по сути является главным циклом работы всего алгоритма. В начале есть некоторый упорядоченный массив однородных многочленов $f_1, ..., f_m$ – то есть исходные многочлены. На шаге i (номер убывает в процессе работы) промежуточным результатом работы алгоритма

является множество G_i – многочлены, образующих базис Грёбнера идеала $\langle f_{m-i+1}, ..., f_m \rangle$.

В начале из обычных многочленов $f_1, ..., f_m$ путём добавления индекса и монома сигнатуры получаются отмеченные многочлены, причём у многочлена f_j моном равен 1, а индекс – j. Начальным состоянием множества G_m является одноэлементное множество $\{m\}$.

Далее, для i от $m-1$ до 1 проводится процедура AlgorithmF5. После всех шагов получается некоторое множество G_1 – множество многочленов, чьи полиномиальные части являются искомым базисом Грёбнера.

5 Остановка алгоритма F5

5.1 S-пары

Определение 7. *Цепь s-пар – последовательность отмеченных многочленов, соседние элементы которой удовлетворяют следующему свойству: старший элемент является s-многочленом младшего элемента и некоторого другого с меньшей сигнатурой. Для соседних элементов вследствие этого свойства выполняется соотношение:* $\mathrm{S}(f_j) = u_j \mathrm{S}(f_{j-1})$.

В дальнейшем для простоты будем называть цепь s-пар просто цепью.

Теорема 1. *Любой отмеченный многочлен может являться начальным элементом лишь конечного числа различных цепей длины 2.*

Для доказательства этой теоремы применяется лемма Диксона: любой мономиальный идеал порождён конечным набором одночленов.

Рассмотрим отмеченный многочлен p с сигнатурой s и упорядоченное по порядку добавления множество $\{p_1, ..., p_i, ...\}$ отмеченных многочленов с сигнатурами, удовлетворяющими условию $\mathrm{S}(p_i) = v_i \mathrm{S}(p)$. Мономиальный идеал из v_i конечно порождён по лемме Диксона, значит, начиная с некоторого номера N все последующие многочлены будут делится на какой-то из предыдущих (для каждого свой).

Но при $k > N$ $\{p, p_k\}$ не может быть цепью, так как $\mathrm{S}(p) \cdot v_i$ перезаписывается $\mathrm{S}(p_j) \cdot \frac{p_i}{p_j}$, а значит, не проходит проверку Rewritten? либо в IsReducible, либо в Spol. Отсюда следует, что есть не более N цепей длины 2, начинающихся с p. ■

Это конечное множество будем называть множеством s-порождённых p.

Теорема 2. *Если алгоритм не останавливается на некоторых выходных данных, то он порождает бесконечную цепь.*

Докажем эту теорему. Многочлен $((1, 1), f_1)$ является началом бесконечного числа различных цепей. Среди конечного числа его порождённых также найдётся элемент, являющийся началом бесконечного числа конечных цепей. Продолжая эту процедуру, получим искомую бесконечную последовательность.■

Если мы рассмотрим два подряд идущих в цепи многочлена p_i, p_{i+1}, то $\mathrm{poly}(p_i)$ не будет меняться после создания p_{i+1} (см. замечание после описания процедуры Reduction). Отсюда можно сделать вывод, что $\mathrm{poly}(p_{i+1}) =$

$c\frac{\mathrm{S}(p_{i+1})}{\mathrm{S}(p_i)}\mathrm{poly}(p_i)+h_i$, где h_i соответствует младшей части s-пары, использованной при созданииp_{i+1} из p_i.

Сформулируем утверждение о наличии сигнатурных редукторов с определёнными свойствами при условии бесконечной работы алгоритма. В дальнейшем это будет приведено к противоречию.

Теорема 3. *Если AlgorithmF5 не останавливается при обработке f_1, то после некоторого шага G_1 содержит пару позиций многочленов f', f, причём f сгенерирован после f' и выполняется:*

- $HM(f')|HM(f)$
- $\dfrac{HM(f')}{\mathrm{S}(f')} > \dfrac{HM(f)}{\mathrm{S}(f)}$.

Для начала распишем (в силу рассуждений перед теоремой):

$$\mathrm{HM}(p_{i+1}) < \mathrm{HM}(\frac{\mathrm{S}(p_{i+1})}{\mathrm{S}(p_i)}p_i) = \mathrm{HM}(h_i)$$

$$\mathrm{S}(p_{i+1}) = \mathrm{S}(\frac{\mathrm{S}(p_{i+1})}{\mathrm{S}(p_i)}p_i) > \mathrm{S}(h_i)$$

Из первого неравенства получаем $\dfrac{\mathrm{HM}(p_i)}{\mathrm{S}(p_i)} > \dfrac{\mathrm{HM}(p_{i+1})}{\mathrm{S}(p_{i+1})}$, поэтому в цепи частные старших мономов и сигнатур строго убывают.

В последовательности $\{\mathrm{HM}(p_i)\}$ по лемме Диксона есть два различных элемента, один из которых делит другой, причём $i < j$ вследствие того, что делимость мономов возможна лишь в случае, когда $\deg(p_i) < \deg(p_j)$, а в цепи степень однородных многочленов не убывает. Отсюда получаем искомое неравенство, поэтому можно взять $f' = p_i, f = p_j$. ■

Обозначим за G_g множество, полученное следующим способом: берём множество $G_1 \cup \mathrm{Done}$ в момент, предшествующий добавлению в Done фиксированного многочлена g с индексом сигнатуры 1 и упорядочиваем в соотвествии с позицией многочленов в R.

Непосредственно из алгоритма можно вывести, что каждая s-пара элементов G_g, сигнатура которых меньше $\mathrm{S}(g)$, удовлетворяет одному из трёх свойств:

1. Одна из частей s-пары была отброшена критерием проверки редуцированности (в IsReducible (условие b.) или CritPair) – *s-пары с частью, удовлетворяющей "критерию F5"*;

2. Одна из частей s-пары была отброшена проверкой Rewritten? (при работе процедур Spol или IsReducible) – *s-пары с частью, удовлетворяющей "критерию перезаписи"*;

3. S-пара не была отброшена, и её s-многочлен был редуцирован по некоторым элементам G_g, а результат был добавлен в G_g – *s-пары с известным G_g-представлением.*

Распространим понятия критерия F5 и критерия перезаписи на произвольные умноженные на моном отмеченные многочлены $tf, f \in G_g$.

Определение 8. *$tf_i, f_i \in G_g$ удовлетворяет критерию F5, если:*

$$\varphi_{index(f_i)+1}(t\mathrm{S}(f_i)) \neq t\mathrm{S}(f_i),$$

где $\varphi_{index(f_i)+1}$ – нормальная форма относительно базиса Грёбнера идеала, полученного на предыдущем шаге алгоритма;

$tf_i, f_i \in G_g$ удовлетворяет критерию перезаписи, если существует j, большее i, такое что $\mathrm{S}(f_j) \mid t\mathrm{S}(f_i)$.

Заметим, что если tf удовлетворяет критерию, то *stp* также ему удовлетворяет (умножение на моном сохраняет критерий).

5.2 Представления

Определение 9. *Представление отмеченного многочлена p в виде*

$$p = \sum_k m_k \cdot b_{i_k}, \qquad b_{i_k} \in G_g$$

с коэффициентами $m_k = c_k t_k \in K \times T$, в которой все пары (t_k, b_{i_k}) различны, называется G_g-представлением p. Члены суммы называются элементами представления.

G_g-представление p – сигнатурное, если для любого номера k выполнено:

$$\mathrm{S}(m_k b_{i_k}) < \mathrm{S}(p).$$

В дальнейшем мы построим упорядоченную последовательность представлений, обладающую некоторыми свойствами, и докажем, что бесконечная работа алгоритма и существование такой последовательности противоречат друг другу.

Для начала надо ввести частичный порядок на элементах представления: $c_i t_i \cdot b_i > c_j t_j \cdot b_j$ тогда и только тогда, когда либо $\mathrm{S}(t_i b_i) > \mathrm{S}(t_j b_j)$, либо они

равны, но $i < j$. Заметим, что если и сигнатуры, и позиции в списке совпадают, то элементы не сравнимы. Но так как в G_g-представлении все пары (t_k, b_{i_k}) различны, то все элементы одного G_g-представления сравнимы между собой. Также в дальнейшем будем заранее предполагать, что представления упорядочены в порядке убывания элементов.

Теперь вводим порядок на представлениях: $\sum_k u_k \cdot v_{i_k} < \sum_k m_k \cdot b_{i_k}$, если первое представление лексикографически меньше второго относительно порядка на элементах представления. Если представления отличаются длиной, то более короткая форма считается меньшей. Так же, как и элементы представления, не все G_g-представления сравнимы: если наибольшие различные элементы отличаются лишь на коэффициент, то представления не сравнимы.

Из введённых порядков на элементах и представлениях ясно, что меньшее представление наследует сигнатурность у большего. Также заметим, что из вполне упорядоченности сигнатур следует вполне упорядоченность элементов представлений, а так как представления сравниваются лексикографически,то представлениях также вполне упорядоченны введённым порядком.

Следующим шагом будет построение конечной последовательности убывающих G_g-представлений для некоторого отмеченного многочлена с сигнатурой меньшей, чем у g. Главным результатом этой секции является следующая

Теорема 4. *Для любого многочлена $mh, m \in K \times T, h \in G_g, \mathrm{S}(mh) < \mathrm{S}(g)$ существует сигнатурное G_g-предствление $mh = \sum_k m_k \cdot b_{i_k}$, удовлетворяющее следующим свойствам при любом k:*

1. *$m_k b_{i_k}$ не удовлетворяет критерию F5;*
2. *$m_k b_{i_k}$ не удовлетворяет критерию перезаписи;*
3. *$HM(m_k b_{i_k}) \leqslant HM(mh)$.*

Идея доказательства: если какой-то элемент не подходит под эти условия, то существует представление этого элемента, которое будет меньше по заданному нами порядку. Тогда, построив такое представление элемента и подставив его вместо самого элемента, получим меньшее представление.

Сформулируем теорему о построении такого представления в различных случаях:

Теорема 5.

1. *Пусть G_g-представление для mh содержит элемент $m_n b_{i_n}$, не удовлетворяющий свойству 1. Тогда у него есть представление, меньшее, чем $m_n b_{i_n} = m_n \cdot b_{i_n}$;*

2. *Пусть G_g-представление для mh содержит элемент $m_n b_{i_n}$ (при этом $\mathrm{S}(mh) < \mathrm{S}(g)$), не удовлетворяющий свойству 2. Тогда у него есть представление, меньшее, чем $m_n b_{i_n} = m_n \cdot b_{i_n}$;*

3. *Если же все элементы G_g-представления для mh (с $\mathrm{S}(mh) < \mathrm{S}(g)$) удовлетворяют свойствам 1 и 2, но есть элемент, не удовлетворяющий свойству 3, то найдется элемент $m_n b_{i_n}$, у которого есть представление, меньшее, чем $m_n b_{i_n} = m_n \cdot b_{i_n}$.*

Важным следствием, которое можно получить из Теоремы 4, является

Теорема 6. *Пусть f – произвольный многочлен. Если существует сигнатурный редуктор $f' \in G_g$ для f, с условием $\mathrm{S}(f')\dfrac{HM(f)}{HM(f')} < \mathrm{S}(g)$, то в G_g есть сигнатурный редуктор для f, который не отбрасывается критерием F5 и критерием перезаписи.*

□ Рассмотрим $mf' = \dfrac{\mathrm{HM}(f)}{\mathrm{HM}(f')} f'$ с сигнатурой меньшей сигнатуры g. По теореме 4 можно найти представление $mf' = \sum\limits_k m_k b_{i_k}$, удовлетворяющее условиям 1-3. По свойству 3, нет элементов с большим старшим мономом, чем у mf', значит для некоторого N выполняется $\mathrm{HM}(m_N b_{i_N}) = \mathrm{HM}(mf') = \mathrm{HM}(f)$. По теореме 4 представление сигнатурно, значит, $\mathrm{S}(m_N b_{i_N}) \leqslant \mathrm{S}(mf') < \mathrm{S}(f)$. Следовательно, $b_{i_N} \in G_g$ – сигнатурный редуктор для f. По свойствам 1-2 он не отбрасывается ни одним из критериев.■

5.3 Доказательство остановки

Теперь наша задача – объединить утверждения из первой и второй секции для получения противоречия. В частности, мы будем использовать результаты теоремы 3 и теоремы 5.

Теорема 7. *Если алгоритм не останавливается на некоторых данных, то на некотором шаге множество $G \cup Done$ будет содержать пару отмеченных многочленов f_1, f, для которых выполняются следующие условия:*

- *f_1 было добавлено в $G \cup Done$ раньше f;*
- *f_1 – сигнатурный редуктор для f;*
- *$\dfrac{HM(f)}{HM(f_1)} f_1$ не удовлетворяет критерию F5 и критерию перезаписи.*

□ Пусть f и f' – многочлены из теоремы 3, а t – отношение их старших мономов. Примем также $g = f.\ tf'$ – сигнатурный редуктор для f, следовательно $t\mathrm{S}(f') < \mathrm{S}(f)$, и по теореме 5 можно найти сигнатурный редуктор $\dfrac{\mathrm{HM}(f)}{\mathrm{HM}(f_1)} f_1$ для f, который не удовлетворяет критерию F5 и критерию перезаписи. ■

Наконец мы готовы доказать основное утверждение этой главы.

Теорема 8. *Алгоритм F5 останавливается на любых входных данных.*

□ Доказываем от противного. Рассмотрим многочлены f_1, f из теоремы 6. f на некотором шаге был добавлен во множество Done. Этому предшествовал вызов процедуры TopReduction, который вернул f как первую половину возвращаемого множества, а это значит, что вызов IsReducible вернул пустое множество. Значит, для всех элементов $G \cup \mathrm{Done}$ хотя бы одно из условий (a) - (d) не выполнилось.

Вспомним условия (a) - (d):

a. В дроби $\dfrac{\mathrm{HT}(f)}{\mathrm{HT}(f_1)}$ числитель делится на знаменатель, то есть эта дробь (обозначим её за u) – корректный моном;

b. $\varphi(ut_j) = ut_j$, где t_j – моном сигнатуры многочлена f_1;

c. Многочлен (u, f_1) не отбрасывается критерием перезаписи;

d. $u\mathrm{S}(f_1) \neq \mathrm{S}(f)$.

Отсюда видно, что теорема 9 обеспечивает выполнения этих условий для f_1: пункты (a) и (c) следует из третьего утверждения теоремы, пункты (b) и (d) – из второго утверждения теоремы.

Таким образом, мы пришли к противоречию. Значит, алгоритм F5 останавливается на любых входных данных. ■.

6 Применение алгоритма F5

Практическое применение для алгоритма $F5$ можно найти, например, в задаче дискретного логарифмирования на эллиптической кривой.

Определение 10. *Эллиптической кривой называется неособая кривая третьего порядка. В подходящих афинных координатах в поле характеристики больше 3 её уравнение можно привести к виду* $y^2 = x^3 + ax + b$.

На эллиптической кривой можно ввести понятие суммы точек (например, [?]).

Определение 11. *Дискретным логарифмом точки* Q *по основанию* P *называется такое число* n, *что* $Q = nP$.

Задача дискретного логарифмирования является одной из ключевых задач теории защиты информации и криптографии. Для любой конечной группы ($|G| = m$) существует тривиальный алгоритм перебора, однако время его работы для больших групп слишком велико, поэтому главным вопросом этой теории является существование субэкспоненциальных или полиномиальных алгоритмов решения задачи дискретного логарифмирования. Если G – группа рациональных точек эллиптической кривой, то существуют алгоритмы, сводящие эту задачу к задаче нахождения дискретного логарифма в некотором конечном поле, однако их сложность все еще достаточно высока. В [Sem2] Семаев предложил новый вероятностный алгоритм решения этой задачи, обладающий субэкспоненциальной сложностью, если $\mathrm{char}\ K = 2$.

6.1 Алгоритм Семаева

Пусть P – точка порядка r в группе $E(F_q)$, где E – эллиптическая кривая над полем F_q, Q – точка на этой кривой и пусть известно, что существует число z, такое что $Q = zP$.

1. Зададим параметр m и подмножество V поля F_q мощности $[q^{\frac{1}{m}}] + 1$.

2. Для случайных целых чисел u, v найдем точку $R = uP + vQ$. Если $R = \mathrm{O}$, то z можно найти из уравнения $uz + v \equiv 0 \pmod r$. В противном случае у R есть аффинные координаты (R_X, R_Y). Если $R_X = x_1 \in V$, то переходим к шагу 6 .

3. При $t = 2, ..., m$ попробуем найти $x_1, ..., x_t \in V$ и $u_1, ..., u_{t-2} \in F_q$, такие что первая из следующих систем из $t - 1$ уравнения удовлетворяется:

$$\begin{cases} S_3(u_1, x_1, x_2) = 0, \\ S_3(u_i, u_{i+1}, x_{i+2}) = 0, \quad 1 \leqslant i \leqslant t-3 \\ S_3(u_{t-2}, x_t, R_X) = 0. \end{cases} \tag{1}$$

При $t = 2$ система состоит из одного уравнения $S_3(x_1, x_2, R_X) = 0$. Если все системы неразрешимы, возвращаемся к шагу 2.

Решения (1) являются решениями уравнения

$$S_{t+1}(x_1, x_2, ..., x_t, R_X) = 0. \tag{2}$$

4. Найдем $y_1, ..., y_t \in F_{q^2}$, такие что

$$(x_1, y_1) + (x_2, y_2) + ... + (x_t, y_t) + uP + vQ = \mathrm{O} \tag{3}$$

5. В работе [Sem1] доказывается, что если некоторые $y_i \in F_{q^2} \setminus F_q$, то сумма точек с такими ординатами является точкой порядка 2 в группе $E(F_q)$.

6. Получаем некоторое линейное уравнение. Таких уравнений надо получить не менее $|V|$, после чего решаем получившуюся линейную систему и находим $z \pmod r$.

Систему из шага 3 можно решать при помощи базиса Грёбнера (метод для этого описан в той же работе [Sem2]).

Пусть

$$\begin{gathered} P_1(x_1, ..., x_n) = 0, \\ P_2(x_1, ..., x_n) = 0, \\ ... \\ P_m(x_1, ..., x_n) = 0. \end{gathered} \tag{4}$$

– система полиномиальных уравнений над полем K.

Определение 12. *Степенью первого падения для системы (4) называется такое наименьшее число d_{ff}, что существуют многочлены $Q_i(x_1, ..., x_n)$, $(1 \leqslant i \leqslant m)$, такие что:*

1. $\max\limits_i(\deg Q_i + \deg P_i) = d_{ff}$,

2. $\deg \sum\limits_i Q_i P_i < d_{ff}$,

3. $\sum\limits_i Q_i P_i \neq 0$.

В работе [PetQuis] было выдвинуто предположение, что степень первого падения системы полиномиальных уравнений не меньше, чем максимальная степень, появляющаяся при решении этой системы при помощи алгоритма F4 нахождения базиса Грёбнера.

В будущем планируется изучить связь степени первого падения системы и максимальной степени, появляющейся при решении системы при помощи алгоритма F5, а также связь между значениями максимальных степеней для разных алгоритмов нахождения базиса Грёбнера.

Отдельной интересной задачей в этом ряду стоит исследование скорости и вероятности решения системы, возникающей на шаге 3 алгоритма Семаева. В частности, существенным результатом было бы нахождение множества эллиптических кривых, для которых алгоритм Семаева при его одновременном использовании с алгоритмом F5 даст субэкспоненциальное время решения задачи дискретного логарифмирования на эллиптической кривой.

Список используемой литературы

[Buch] Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nuildimensionalen Polynomideal*, Dissertation, Universität Innsbruck, 1965.

[Faug1] Faugère, J.-C. *A new efficient algorithm for computing Gröbner bases (F4)*, Journal of Pure and Applied Algebra, Volume 139, Issues 1–3, 1999, Pages 61-88, ISSN 0022-4049.

[Faug2] Faugere, J.-C., *A new efficient algorithm for computing Groebner bases without reduction to zero (F_5)*, in: ISSAC '02: Proceedings from the International Symposium on Symbolic and Algebraic Computation (2002), pp. 75-83.

[Steg] Stegers, T., *Faugere's F5 Algorithm Revisited*, Thesis for the degree of Diplom-Mathematiker, Darmstadt, 2005.

[Ger] О. Н. Герман, *Доказательство критерия Фожера для алгоритма F5*, Матем. заметки, 88:4 (2010), 502–510; Math. Notes, 88:4 (2010), 479–486.

[Gal] В. В. Галкин, *Остановка алгоритма F*5, Программирование, 2014, №2.

[EdPer] Eder C., Perry J., *F5C: A variant of Faugere's F5 algorithm with reduced Groebner bases*, J. Symb. Comput., 2010, V. 45, №12. P. 1442-1458.

[OstrZfas] Острик, В., Цфасман, М., *Алгебраическая геометрия и теория чисел: рациональные и эллиптические кривые*, МЦНМО, ISBN 978-5-94057-789-8, 2001

[Sem1] Semaev, I. *Summation polynomials and the discrete logarithm problem on elliptic curves*, Cryptology ePrint Archive 2004/031, 2004.

[Sem2] Semaev, I. *New algorithm for the discrete logarithm problem on elliptic curves*, Cryptology ePrint Archive 2015/310, 2015.

[PetQuis] Petit, C., Quisquater, J. *On polynomial system arising from a Weil descent*, in ASIACRYPT 2012, LNCS, 7658, pp. 451-466, Springer 2012.