Diplomarbeit

Das Kryptosystem HFE und quadratische Gleichungssysteme über endlichen Körpern

Magnus Daum

angefertigt
im Studiengang Mathematik
am Fachbereich Mathematik
der Universität Dortmund

Dortmund, August 2001
Betreuer: Prof. Dr. E. Becker

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein großer Teil der heutzutage in der Praxis eingesetzten, asymmetrischen Kryptographieverfahren basiert auf Berechnungen modulo großer, ganzer Zahlen, wie beispielsweise das wohl bekannteste und am weitesten verbreitete Public-Key-Kryptosystem RSA.

Die Sicherheit solcher Verfahren basiert meist auf der Annahme der Schwierigkeit eines der beiden folgenden Probleme: des Faktorisierens großer Zahlen oder des Berechnens diskreter Logarithmen modulo einer Primzahl.
Diese Annahmen gelten zwar als gesichert, sind aber bisher nicht bewiesen. Daher ist es trotz des guten Rufs solcher Verfahren wie RSA durchaus sinnvoll, nach alternativen Public-Key-Kryptoverfahren zu suchen, deren Sicherheit nicht auf diesen Annahmen beruht.

Eine Klasse solcher, alternativer Public-Key-Verfahren stellen die multivariaten Kryptoverfahren dar. Dies sind Verfahren, die mit multivariaten, polynomialen Gleichungssystemen über endlichen Körpern arbeiten. Das 1996 von Jacques Patarin vorgestellte Kryptosystem HFE gilt heute als eines der stärksten Kryptosysteme dieser Art:

> „*The RSA public key cryptosystem is based on a single modular equation in one variable. A natural generalization of this approach is to consider systems of several modular equations in several variables (...)*
>
> *HFE (...) is believed to be one of the strongest schemes of this type.*“
>
> Aviad Kipnis, Adi Shamir (in [KS99])

Multivariate Kryptosysteme wie HFE sind allgemein nicht so verbreitet und von daher auch nicht so gut untersucht wie die oben erwähnten Systeme. Dies führt natürlich dazu, dass das Vertrauen in die Sicherheit nicht so groß sein kann wie bei Verfahren, die lange bekannt sind und viel untersucht wurden (wie beispielsweise RSA seit mehr als 20 Jahren). Zudem kann es durchaus auch als Nachteil der multivariaten Systeme angesehen

werden, dass ihre Sicherheit nicht auf einem wohlbekannten, gut studierten Problem wie dem Faktorisieren beruht, vor allem da manche multivariaten Kryptosysteme, wie Patarin schreibt, „auf spektakuläre Art gebrochen wurden“.

Andererseits haben multivariate Kryptosysteme neben der Unabhängigkeit von den üblichen, in vielen Verfahren verwendeten Annahmen (wie der Schwierigkeit des Faktorisierens oder des Berechnens diskreter Logarithmen) noch weitere Vorteile:
Beispielsweise können mit HFE sehr kurze, digitale Signaturen erzeugt werden. So kann in Bezug auf die bekannten Attacken eine HFE-Signatur von ungefähr 128 Bit als durchaus sicher angesehen werden. Dagegen sind für RSA schon 512 Bit nicht mehr als ausreichend zu betrachten, nachdem auf der Eurocrypt 2000 die Faktorisierung eines 512 Bit RSA-Moduls vorgestellt wurde.
Ein weiterer Vorteil multivariater Kryptosysteme ist ihre gute Eignung für die Implementierung auf Smartcards, da diese Systeme häufig auf einfachen Berechnungen basieren, die zudem auch nur wenig RAM benötigen.

Angefangen hat die Entwicklung der multivariaten Kryptographie schon in den 80er Jahren. Allerdings wurden die ersten damals vorgestellten Systeme dieser Art schnell gebrochen, so dass zunächst keine weiteren Versuche durchgeführt wurden, solche Systeme zu entwickeln.
Auf der Eurocrypt 1988 wurde dann von Hideki Imai und Tsutomu Matsumoto das erste erfolgversprechende System C^* vorgestellt. Dieses Verfahren wurde erst 1995 von Jacques Patarin gebrochen.
Ein Jahr später hat Patarin dann auf der Eurocrypt 1996 das Verfahren HFE als eine mögliche Verallgemeinerung von C^* vorgestellt. Durch diese Verallgemeinerung hat er versucht, die Schwächen zu beseitigen, die es ermöglichten, C^* zu brechen.
Bis heute gilt HFE als nicht gebrochen. Denn auch die schnellsten bislang bekannten Attacken wie etwa die Attacken von Nicolas Courtois oder von Aviad Kipnis und Adi Shamir sind nicht deutlich schneller als die triviale Attacke über eine vollständige Suche, so dass diese Attacken leicht durch eine geeignete Parameterwahl verhindert werden können. Für bestimmte Varianten von HFE, die sogenannten Perturbationen, die durch leichte Modifikationen des ursprünglichen Systems entstehen, sind bislang gar keine Attacken bekannt, die schneller als eine vollständige Suche wären.

1.1 Überblick

Das Ziel dieser Arbeit ist es, das Kryptoverfahren HFE vor allem in Bezug auf seine mathematischen Eigenschaften vorzustellen. Da die Sicherheit von HFE stark mit dem Lösen von quadratischen Gleichungssystemen zusammenhängt, sollen zudem dieses Problem näher untersucht und die Auswirkungen auf HFE dargestellt werden.

In den weiteren Abschnitten dieses einleitenden Kapitels werden zunächst kurz die in dieser Arbeit verwendeten Notationen erläutert. Danach werden Übersichten über die Grundlagen zweier Bereiche gegeben, die für die weitere Arbeit benötigt werden: Abbildungen zwischen endlichen Körpern und die Komplexitätstheorie.

Im zweiten Kapitel wird dann HFE vorgestellt. Dazu wird zunächst das zugrunde liegende Schema beschrieben. Da allerdings die bei HFE verwendete Funktion nicht wie üblich bijektiv ist, müssen spezielle Vorkehrungen getroffen werden, um HFE zum Verschlüsseln und Signieren einsetzen zu können. Diese werden anschließend beschrieben.
Danach werden einige Eigenschaften wie beispielsweise die Wirksamkeit dieser Vorkehrungen näher betrachtet und es wird untersucht, welche Auswirkungen diese Eigenschaften auf die Wahl der Parameter haben. Zudem werden in diesem Kapitel noch sogenannte Perturbationen — gewisse Varianten von HFE — vorgestellt und die wichtigsten Sicherheitsgrundlagen von HFE vor allem aus komplexitätstheoretischer Sicht erläutert.

Da ein wichtiger Sicherheitsaspekt von HFE die Schwierigkeit des Problems MQ, also des Lösens von quadratischen Gleichungssystemen ist, wird dieses Problem in den folgenden Kapiteln näher betrachtet. In Kapitel 3 wird dazu zunächst beschrieben, wie allgemein polynomiale Gleichungssysteme mit Hilfe einer Gröbnerbasis gelöst werden können. Insbesondere wird dabei natürlich der Spezialfall von HFE-Systemen betrachtet. Zu dessen näherer Untersuchung wurden auch praktische Simulationen durchgeführt, die ebenfalls in diesem Kapitel analysiert werden.
In Kapitel 4 wird dann ein weiterer Spezialfall von MQ untersucht: der Fall unter- und überbestimmter Gleichungssysteme. Es werden einige Algorithmen vorgestellt, mit denen besonders stark unter- oder überbestimmte Gleichungssysteme effizienter gelöst werden können. Diese Spezialfälle sind in Bezug auf HFE in zweierlei Hinsicht interessant:
Zum einen können bei den im zweiten Kapitel beschriebenen Perturbationen über- und unterbestimmte Gleichungssysteme auftreten und zum anderen finden Algorithmen für überbestimmte Gleichungssysteme auch Anwendung in der sogenannten Attacke über MinRank.
Der Ablauf dieser Attacke und der Ablauf der sogenannten Attacke über implizite Gleichungen wird im abschließenden Kapitel 5 zusammengefasst. Diese beiden Attacken sind die zwei effizientesten, momentan bekannten Arten von Attacken auf HFE.

Zusätzlich ist dieser Arbeit eine CD beigelegt, die neben einer elektronischen Fassung dieser Arbeit, die Programme und die vollständigen Ergebnisse zu den Simulationen enthält.

Die Abschnitte dieser Arbeit, die sich direkt mit HFE beschäftigen, basieren neben eigenen Untersuchungen im Wesentlichen auf den in [CGP99] zusammengefassten Texten und den

später veröffentlichten Artikeln [CKPS00] und [Cou01a]. Insbesondere dem Kapitel über Gröbnerbasen liegt allerdings noch zahlreiche andere Literatur zugrunde. Ein Verzeichnis der gesamten bei der Erstellung dieser Arbeit verwendeten Literatur ist am Ende der Arbeit zu finden.

1.2 Notationen und Bezeichnungen

In dieser Arbeit werden die folgenden Notationen und Bezeichnungen verwendet:

$\mathbb{F}_q$	der endliche Körper mit q Elementen
$\overline{K}$	der algebraische Abschluss eines Körpers K
$\mathrm{grad}_x(p)$	der Grad des Polynoms p als Polynom in x aufgefasst
$K^{m\times n}$	die Menge der Matrizen mit m Zeilen und n Spalten mit Einträgen aus K
$\langle U\rangle_K$	der von der Menge U erzeugte K-Vektorraum
$\dim_K(V)$	die Dimension von V als K-Vektorraum
$\lvert M\rvert$	die Kardinalität der Menge M
$\lfloor a\rfloor$	die größte, ganze Zahl $\leq a$
$\lceil a\rceil$	die kleinste, ganze Zahl $\geq a$
$\Pr(A)$	die Wahrscheinlichkeit, dass das Ereignis A eintritt
$\Pr(A \mid B)$	die bedingte Wahrscheinlichkeit für das Eintreten des Ereignisses A unter der Bedingung, dass B eingetreten ist
S_d	die symmetrische Gruppe auf $\{1,\ldots,d\}$
$\sim$	Äquivalenzrelation auf den d-Tupeln aus $\{1,\ldots,n\}^d$, definiert durch $(i_1,\ldots,i_d)\sim(j_1,\ldots,j_d) :\Leftrightarrow \exists\sigma\in S_d : i_k = j_{\sigma(k)}$ für $k=1,\ldots,d$
$x \parallel y$	die Konkatenation von x und y, d.h. für $x=(x_1,\ldots,x_n)$ und $y=(y_1,\ldots,y_r)$ ist $x\parallel y=(x_1,\ldots,x_n,y_1,\ldots,y_r)$
##	leitet Kommentare in Algorithmen ein
$U(\varphi)$	die univariate Darstellung der Funktion φ (vgl. Definition 1.3.3 *i*))
$M(\varphi)$	die multivariate Darstellung der Funktion φ (vgl. Definition 1.3.3 *ii*))
$\mathcal{U}_{n,d}$	die Menge der univariaten Darstellungen, bei denen die auftretenden Exponenten das q-Gewicht d nicht überschreiten (vgl. Definition 1.3.7)
$\mathcal{M}_{n,d}$	die Menge der multivariaten Darstellungen mit n Polynomen in n Variablen vom Grad d (vgl. Definition 1.3.7)
HFEP	das HFE-Problem (vgl. Problem 2.5.1)
MQ	das Problem, multivariate, quadratische Gleichungssysteme zu lösen (vgl. Problem 2.5.2)
IP	das Problem Isomorphism of Polynomials (vgl. Problem 2.5.3)
MinRank	das Problem MinRank (vgl. Problem 2.5.4)

1.3 Grundlagen

1.3.1 Darstellungen von Abbildungen zwischen endlichen Körpern

Die Idee des HFE-Kryptosystems basiert darauf, dass Abbildungen zwischen endlichen Körpern auf verschiedene Arten dargestellt werden können. Aus diesem Grund werden in diesem Abschnitt zunächst einige Grundlagen über die für HFE wichtigen, univariaten und multivariaten Darstellungen polynomialer Abbildungen über endlichen Körpern beschrieben.

Um die Notationen zu vereinfachen seien in diesem und auch in den folgenden Kapiteln — soweit nicht anders angegeben — folgende Schreibweisen vereinbart:
q sei eine Primzahlpotenz, $n \in \mathbb{N}$ eine natürliche Zahl, $K = \mathbb{F}_q$ der endliche Körper mit q Elementen und $L = \mathbb{F}_{q^n}$ der (bis auf Isomorphie) eindeutig bestimmte Erweiterungskörper über K vom Grad n.
Da L als n-dimensionaler K-Vektorraum aufgefasst werden kann, sei ferner ein K-Isomorphismus $\phi : L \to K^n$ fest vorgegeben, etwa nach Wahl einer K-Basis $\gamma_1, \ldots, \gamma_n$ von L durch

$$\phi : \begin{array}{rcl} L & \to & K^n \\ a & \mapsto & (a_1, \ldots, a_n), \text{ wobei } a = \sum_{i=1}^{n} a_i \gamma_i. \end{array}$$

Mit Hilfe von ϕ können dann Abbildungen von L bzw. K^n in sich auch als Abbildungen des jeweils anderen Raumes aufgefasst werden. Dazu seien für $\varphi : L \to L$ bzw. $\varphi' : K^n \to K^n$ die jeweils zugehörigen Abbildungen bezeichnet mit

$$\begin{array}{rcl} \varphi_{K^n} & := & \phi \circ \varphi \circ \phi^{-1} \\ \text{bzw.} \quad \varphi'_L & := & \phi^{-1} \circ \varphi' \circ \phi. \end{array}$$

Das erste Lemma hilft, Abbildungen zwischen endlichen Körpern mit Polynomen von beschränktem Grad zu identifizieren:

Lemma 1.3.1
Sei $\varphi : K^n \to K$ eine beliebige Abbildung. Dann existiert ein eindeutig bestimmtes Polynom $p \in K[x_1, \ldots, x_n]$ mit $grad_{x_i}(p) < |K| = q$ für alle $i = 1, \ldots, n$, für das gilt

$$\varphi(a) = p(a) \qquad \text{für alle } a \in K^n.$$

Beweis:
Wähle als p das Interpolationspolynom

$$p(x_1, \ldots, x_n) = \sum_{c=(c_1,\ldots,c_n) \in K^n} \frac{\prod_{i=1}^{n} \prod_{b \in K \setminus \{c_i\}} (x_i - b)}{\prod_{i=1}^{n} \prod_{b \in K \setminus \{c_i\}} (c_i - b)} \cdot \varphi(c).$$

Dieses hat $grad_{x_i}(p) \leq q - 1$ und es gilt offensichtlich $p(a) = \varphi(a)$ für alle $a \in K^n$.

Zur Eindeutigkeit:
Die Polynome $p \in K[x_1, \ldots, x_n]$ mit $grad_{x_i}(p) < q$ für alle $i = 1, \ldots, n$ sind gerade die Polynome der Form

$$p(x_1, \ldots, x_n) = \sum_{i_1=0}^{q-1} \ldots \sum_{i_n=0}^{q-1} a_{i_1 \ldots i_n} \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \quad \text{mit } a_{i_1 \ldots i_n} \in K.$$

Es gibt also genau $|K|^{(q^n)} = q^{(q^n)}$ solche Polynome.
Ferner gibt es für zwei endliche Mengen X und Y aber auch genau $|Y|^{|X|}$ Abbildungen von X nach Y, also gerade $|K|^{|K^n|} = q^{(q^n)}$ Abbildungen $\varphi : K^n \to K$.
Zu jeder Abbildung kann also nur höchstens ein solches Polynom existieren, welches somit eindeutig bestimmt ist. ■

Damit können Abbildungen $\varphi : L \to L$ bzw. $\varphi' : K^n \to K^n$ mit Hilfe der oben vorgestellten Identifikationen $\varphi_{K^n} := \phi \circ \varphi \circ \phi^{-1}$ und $\varphi'_L := \phi^{-1} \circ \varphi' \circ \phi$ auf zwei verschiedene Weisen jeweils eindeutig dargestellt werden, wie der folgende Satz zeigt:

Satz 1.3.2
Sei $\varphi : L \to L$ (oder $\varphi : K^n \to K^n$) eine beliebige Abbildung. Dann kann φ dargestellt werden durch

i) ein univariates Polynom $f \in L[x]$ mit $grad(f) \leq q^n - 1$
*(**univariate Darstellung**),*

ii) n Polynome $p_1, \ldots, p_n \in K[x_1, \ldots, x_n]$ mit $grad_{x_i}(p_j) \leq q-1$ für alle $i, j = 1, \ldots, n$
*(**multivariate Darstellung**).*

Beide Darstellungen sind (nach Wahl des Isomorphismus ϕ) eindeutig.

Beweis:
Hier nur der Beweis für $\varphi : L \to L$, der Beweis für $\varphi : K^n \to K^n$ folgt analog durch Betrachtung von φ_L.
i) folgt direkt aus Lemma 1.3.1 mit $n = 1$ durch Ersetzen von K durch L.
zu *ii*): Betrachte die durch ϕ zu φ gegebene Abbildung $\widetilde{\varphi} := \varphi_{K^n} : K^n \to K^n$ mit $\widetilde{\varphi}(x_1, \ldots, x_n) = (\widetilde{\varphi}_1(x_1, \ldots, x_n), \ldots, \widetilde{\varphi}_n(x_1, \ldots, x_n))$. Jedes $\widetilde{\varphi}_j$ ist also eine eindeutig bestimmte Abbildung $K^n \to K$, die nach Lemma 1.3.1 gerade durch ein eindeutig bestimmtes Polynom p_j mit $grad_{x_i}(p_j) \leq q - 1$ dargestellt werden kann. ■

Definition 1.3.3
Zu einer Abbildung $\varphi : L \to L$ oder auch $\varphi : K^n \to K^n$ und einem Isomorphismus $\phi : L \to K^n$ bezeichne

i) $U(\varphi)$ die univariate Darstellung f und

ii) $M(\varphi)$ die multivariate Darstellung $(p_1, \ldots, p_n)$

aus Satz 1.3.2, die beide nach diesem Satz eindeutig bestimmt sind.

Ziel des restlichen Abschnitts ist es, das Verhältnis dieser beiden Darstellungen zueinander, insbesondere das Verhältnis der Grade der Polynome in $M(\varphi)$ zur Darstellungsdichte von $U(\varphi)$ zu beschreiben.

Für *lineare* Abbildungen $\varphi : K^n \to K^n, (x_1, \ldots, x_n) \mapsto A \cdot (x_1, \ldots, x_n)^t$ mit $A \in K^{n \times n}$ ist die Situation klar:

Satz 1.3.4
Die K-linearen Abbildungen

$$\left\{\varphi : K^n \to K^n \mid \varphi(x_1, \ldots, x_n) = A \cdot (x_1, \ldots, x_n)^t \ , \ A \in K^{n \times n}\right\}$$

können gerade beschrieben werden durch die multivariaten Darstellungen der Form

$$M(\varphi) = (p_1, \ldots, p_n) \ \textit{mit } (p_i \textit{ ist homogen vom Grad } 1 \textit{ oder } p_i = 0) \textit{ für } i = 1, \ldots, n$$

und durch die univariaten Darstellungen der Form

$$U(\varphi) = \sum_{i=0}^{n-1} a_i \cdot x^{q^i} \ \textit{mit } a_i \in L.$$

Beweis:
Die multivariate Darstellung folgt direkt durch Ausmultiplizieren der Matrixdarstellung.
Zur univariaten Darstellung:
Der Frobenius-Homomorphismus $x \mapsto x^q$ und damit auch seine mehrfache Hintereinanderausführung $x \mapsto x^{q^i}$ sind bekanntermaßen K-linear. Damit folgt aber sofort aufgrund der Zusammensetzung durch Additionen und skalare Multiplikationen, dass auch die durch $U(\varphi)$ beschriebene Abbildung $x \mapsto \sum_{i=0}^{n-1} a_i \cdot x^{q^i}$ K-linear ist.
Ferner gibt es sowohl genau $q^{\binom{n}{2}}$ lineare Abbildungen $(x_1, \ldots, x_n) \mapsto A \cdot (x_1, \ldots, x_n)^t$ mit $A \in K^{n \times n}$ als auch genau $(q^n)^n = q^{n^2}$ Polynome der Form $\sum_{i=0}^{n-1} a_i \cdot x^{q^i}$ mit $a_i \in L$. Da nach Satz 1.3.2 diese Darstellungen eindeutig sind, folgt die Behauptung. ■

K-lineare Abbildungen werden über L also durch die sogenannten *linearisierten Polynome* $\sum_{i=0}^{n-1} a_i \cdot x^{q^i}$ eindeutig dargestellt.

Um zu beschreiben, wie sich Abbildungen höheren Grades verhalten, werden zunächst noch einige Begriffe eingeführt:

Definition 1.3.5
Zu $k \in \mathbb{N}_0$ mit $k = \sum_i k_i \cdot q^i$, $k_i \in \{0, \ldots, q-1\}$ bezeichne

$$G_q(k) := \sum_i k_i$$

*das q-**Gewicht** von k.*

Bemerkung 1.3.6
Für das q-Gewicht der Summe zweier Zahlen $k, l \in \mathbb{N}_0$ gilt

$$G_q(k+l) \leq G_q(k) + G_q(l).$$

Nun können die univariaten und multivariaten Darstellungen nach der Dichte ihrer Darstellung bzw. nach dem Grad der vorkommenden Polynome unterschieden werden:

Definition 1.3.7

- *Die Menge der Polynomsysteme mit n Polynomen und n Variablen vom Grad $\leq d$, die multivariate Darstellungen $M(\varphi)$ von Abbildungen $\varphi : L \to L$ beschreiben, wird mit*

$$\mathcal{M}_{n,d} := \left\{ (p_1, \ldots, p_n) \in (K[x_1, \ldots, x_n])^n \;\middle|\; \begin{array}{l} grad(p_j) \leq d \quad und \\ grad_{x_i}(p_j) \leq q-1 \end{array} \quad \text{für alle } i, j \right\}$$

  bezeichnet.

- *Die Menge der univariaten Darstellungen $U(\varphi)$ von Abbildungen $\varphi : L \to L$ wird unterteilt in*

$$\mathcal{U}_{n,d} := \left\{ \sum_{i=0}^{q^n-1} a_i \cdot x^i \in L[x] \;\middle|\; a_i \neq 0 \text{ nur für } G_q(i) \leq d \right\}.$$

Bemerkung 1.3.8
Offensichtlich gilt für $d_1 \leq d_2$

$$\mathcal{M}_{n,d_1} \subseteq \mathcal{M}_{n,d_2} \quad und \quad \mathcal{U}_{n,d_1} \subseteq \mathcal{U}_{n,d_2}$$

und ferner folgt aus Bemerkung 1.3.6 direkt

$$p \in \mathcal{U}_{n,d_1},\ p' \in \mathcal{U}_{n,d_2} \quad \Rightarrow \quad pp' \in \mathcal{U}_{n,d_1+d_2}.$$

Der folgende Satz zeigt, dass sich diese Einteilungen der univariaten und der multivariaten Darstellungen entsprechen:

Satz 1.3.9
Für beliebige $n, d \in \mathbb{N}$ gilt

$$\{\varphi : L \to L \mid U(\varphi) \in \mathcal{U}_{n,d}\} \quad = \quad \{\varphi : L \to L \mid M(\varphi) \in \mathcal{M}_{n,d}\}.$$

Beweis:
„ $\subseteq$ “ :
Da $\mathcal{M}_{n,d}$ additiv abgeschlossen ist, genügt es, monomiale Abbildungen $\varphi : L \to L$ mit $U(\varphi) = b \cdot x^i$, $b \in L$ und $i \in \{0, \ldots, n-1\}$ zu betrachten. Sei also $\varphi : L \to L$ mit

$$U(\varphi) = b \cdot x^{q^{i_1}+q^{i_2}+\ldots+q^{i_{d'}}} = b \cdot x^{q^{i_1}} \cdot \ldots \cdot x^{q^{i_{d'}}} \quad \text{mit } b \in L \text{ und } d' \leq d.$$

Da $x \mapsto x^{q^k}$ eine K-lineare Abbildung ist, gilt für alle $a = \sum a_i \gamma_i \in L$ (mit $a_i \in K$, $\gamma_1, \ldots, \gamma_n$ die K-Basis von L, die den Isomorphismus ϕ beschreibt)

$$a^{q^k} = \left(\sum_i a_i \cdot \gamma_i \right)^{q^k} = \sum_i a_i \cdot (\gamma_i)^{q^k} = \sum_i a_i \cdot \sum_j \lambda_{ij} \gamma_j = \sum_{i,j} \lambda_{ij} a_i \cdot \gamma_j$$

für gewisse $\lambda_{ij} \in K$. Die Abbildung φ kann mit $(x_1, \ldots, x_n) = \phi(x)$ also auch geschrieben werden als

$$\begin{aligned} \varphi(x) &= b \cdot x^{q^{i_1}} \cdot \ldots \cdot x^{q^{i_{d'}}} \\ &= b \cdot \left(\sum_{i,j} \lambda_{ij}^{(i_1)} x_i \cdot \gamma_j \right) \cdot \ldots \cdot \left(\sum_{i,j} \lambda_{ij}^{(i_{d'})} x_i \cdot \gamma_j \right). \end{aligned}$$

Ausmultipliziert und neu in der Basis geschrieben ergibt dies

$$\varphi(x) = \sum_i p_i(x_1, \ldots, x_n) \cdot \gamma_i$$

mit Polynomen $p_i \in K[x_1, \ldots, x_n]$ mit $grad(p_i) \leq d' \leq d$. Dabei kann $grad_{x_j}(p_i) \leq q-1$ erreicht werden, indem solange alle vorkommenden x_j^q durch x_j ersetzt werden, bis die Grade klein genug sind, was an der Abbildung, die auf $K = \mathbb{F}_q$ definiert ist, nichts ändert. Damit ist also

$$M(\varphi) = (p_1, \ldots, p_n) \in \mathcal{M}_{n,d}.$$

„ $\supseteq$ “ :
Da auch $\mathcal{U}_{n,d}$ additiv abgeschlossen ist, genügt es, nur gewisse Abbildungen $\varphi : L \to L$ zu betrachten, nämlich die mit $M(\varphi) = (p_1, \ldots, p_n) \in \mathcal{M}_{n,d}$, so dass

$$\begin{aligned} p_j &= b \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}, \quad \text{mit } b \in K,\ i_k \leq q-1 \text{ und } \sum i_k \leq d \\ \text{und } p_i &= 0 \text{ für } i \neq j. \end{aligned}$$

Nach Satz 1.3.4 existiert zu jeder der linearen Abbildungen

$$\pi_k : K^n \to K^n, \quad (x_1, \ldots, x_n) \mapsto (x_k, 0, \ldots, 0)$$

eine univariate Darstellung

$$U(\pi_k) = \sum_{i=0}^{n-1} a_i \cdot x^{q^i} \in L[x], \text{ also } U(\pi_k) \in \mathcal{U}_{n,1}.$$

Sei nun ohne Einschränkung der Isomorphismus ϕ so gewählt, dass das erste Basiselement $\gamma_1 = 1 \in L$ ist. Dann ist nämlich für $l \in \{0, \ldots, q-1\}$ auch die Abbildung

$$(\pi_k)^l : K^n \to K^n, \quad (x_1, \ldots, x_n) \mapsto \left((x_k)^l, 0, \ldots, 0 \right)$$

direkt darstellbar durch das univariate Polynom

$$U\left(\left(\pi_k\right)^l\right) = \left(U\left(\pi_k\right)\right)^l,$$

da die Multiplikation an der ersten Stelle in K^n wegen $\gamma_1 = 1$ gerade der Multiplikation in L entspricht. Damit ist es nun leicht zu sehen, dass die oben gewählte Funktion φ dargestellt wird durch das univariate Polynom

$$U\left(\varphi\right) = b \cdot \left(U\left(\pi_1\right)\right)^{i_1} \cdot \ldots \cdot \left(U\left(\pi_n\right)\right)^{i_n} \cdot \gamma_j$$

wobei sich durch die Multiplikation mit γ_j gerade die Verschiebung an die j-te Stelle in K^n ergibt. Nach Bemerkung 1.3.8 folgt wegen $\pi_k \in \mathcal{U}_{n,1}$ auch

$$\left(U\left(\pi_k\right)\right)^{i_k} \in \mathcal{U}_{n,i_k}$$

und wegen $\sum i_k \leq d$ weiter

$$U\left(\varphi\right) \in \mathcal{U}_{n,\sum i_k} \subseteq \mathcal{U}_{n,d}.$$

■

1.3.2 Komplexitätstheorie

In diesem Abschnitt werden kurz die grundlegenden Begriffe aus dem Bereich der Komplexitätstheorie erläutert, die in dieser Arbeit verwendet werden. Die hier verwendeten Definitionen richten sich im Wesentlichen nach der Einführung in die theoretische Informatik von Ingo Wegener (siehe [Weg99]), in der auch die Beweise der hier beschriebenen Sätze zu finden sind.

In der Kryptographie ist es vor allem wichtig zu zeigen, dass manche Probleme nicht effizient lösbar sind. Deshalb soll dieser Begriff zunächst formalisiert werden:

Definition 1.3.10
Die Klasse $\mathcal{P}$ ist die Klasse aller Probleme, für die es eine (deterministische) Turingmaschine gibt, deren worst case Rechenzeit polynomiell beschränkt ist. Ein Problem, das in $\mathcal{P}$ liegt, wird als ***effizient lösbar*** *bezeichnet.*

Für die genaue Definition einer Turingmaschine siehe [Weg99].

Die worst case Rechenzeit einer Turingmaschine M wird zu jedem $n \in \mathbb{N}$ definiert als die maximale Rechenzeit, die M auf Eingaben der Länge n benötigt. Ein Problem wird in der Komplexitätstheorie also schon dann nicht mehr als effizient lösbar bezeichnet, wenn es einzelne Instanzen dieses Problems gibt, die große Rechenzeit benötigen (vgl. auch Bemerkung 1.3.25).

Der in Definition 1.3.10 festgelegte Effizienzbegriff hängt natürlich sehr stark vom Begriff der Turingmaschine ab. Er macht aber trotzdem auch im allgemeinen, praktischen Bezug Sinn, da sich nach der (verallgemeinerten) Churchschen These (vgl. [Weg99]) die praktischen Rechenzeiten beliebiger Implementierungen von den theoretischen Rechenzeiten, die Turingmaschinen benötigen, nur um polynomielle Faktoren unterscheiden.

Um zu zeigen, dass ein Problem in $\mathcal{P}$ liegt, genügt es also, einen Algorithmus anzugeben, dessen worst case Laufzeit polynomiell beschränkt ist.
Dagegen ist es schwieriger zu zeigen, dass ein Problem nicht in $\mathcal{P}$ liegt. Dazu werden zunächst nur sogenannte Entscheidungsprobleme, d.h. Probleme, die nur die Ausgaben „Ja“ (bzw. 1) oder „Nein“ (bzw. 0) zulassen, betrachtet:

Definition 1.3.11
Sei Σ ein Eingabealphabet und Σ^* die Menge aller Wörter (d.h. aller endlichen Strings) über diesem Alphabet.
*Dann kann ein **Entscheidungsproblem** durch eine Funktion $f : \Sigma^* \to \{0, 1\}$ bzw. durch die dazugehörige **Sprache** $L := f^{-1}(1) \subseteq \Sigma^*$ dargestellt werden. In den folgenden Sätzen und Definitionen werden das Entscheidungsproblem und die zugehörige Sprache miteinander identifiziert.*

Bei solchen Entscheidungsproblemen (beispielsweise etwa „Gibt es eine Lösung des Gleichungssystems $\mathcal{S}$?“) kann oft folgender Effekt beobachtet werden: Es ist unter Umständen schwierig, das Problem zu lösen (also etwa zu entscheiden, ob eine Lösung existiert). Aber durch glückliches Raten könnte, falls die Antwort „Ja“ lauten muss, ein Beweis dafür (im Beispiel also eine Lösung) gefunden werden, der in polynomieller Zeit verifiziert werden kann. Ein solches Vorgehen wird durch (rein hypothetische) sogenannte nichtdeterministische Turingmaschinen formalisiert.

Eine genaue Definition einer nichtdeterministischen Turingmaschine ist ebenfalls in [Weg99] zu finden. Hier genügt es, zu wissen, dass eine nichtdeterministische Turingmaschine in jedem Schritt zwischen verschiedenen möglichen Operationen wählen kann und so auch zu derselben Eingabe verschiedene Ausgaben produzieren kann. Falls es zu einer Eingabe $x \in \Sigma^*$ mindestens einen Rechenweg mit der Ausgabe „Ja“ gibt, dann ist die Vorstellung von einer nichtdeterministischen Turingmaschine, dass sie sich durch glückliches Raten in jedem Schritt für die Aktion entscheidet, die zum kürzesten solchen Rechenweg gehört.

Damit kann die oben informal beschriebene Klasse $\mathcal{NP}$ von Entscheidungsproblemen, die durch glückliches Raten effizient lösbar wären, formal erfasst werden:

Definition 1.3.12

- *Eine nichtdeterministische Turingsmaschine* ***akzeptiert*** *eine Sprache L (bzw. löst das zugehörige Entscheidungsproblem), falls für alle $x \in \Sigma^*$ gilt, dass es zur Eingabe x einen Rechenweg mit Ausgabe „Ja“ genau dann gibt, wenn $x \in L$ ist.*
  Für $x \in L$ sei die Laufzeit dann die Länge des kürzesten solchen Rechenweges, für $x \notin L$ sei die Laufzeit 0.

- *Die* ***worst case Laufzeit*** *sei analog zum deterministischen Fall die maximale Laufzeit für Eingaben einer gewissen Länge.*

- *$\mathcal{NP}$ bezeichne die Klasse der Entscheidungsprobleme bzw. Sprachen, für die es eine nichtdeterministische Turingmaschine gibt, die sie in polynomiell beschränkter worst case Laufzeit akzeptiert.*

- *co-$\mathcal{NP}$ bezeichne die Klasse der Sprachen $\overline{L} := \Sigma^* \backslash L$, für die L in $\mathcal{NP}$ liegt.*

Bemerkung 1.3.13
$\mathcal{NP}$ (bzw. co-$\mathcal{NP}$) enthält genau die Entscheidungsprobleme, für die (wie oben beschrieben) durch glückliches Raten ein Beweis für die Antwort „Ja“ (bzw. für die Antwort „Nein“ im Falle von co-$\mathcal{NP}$) gefunden werden könnte, der in polynomieller Zeit verifiziert werden kann (vgl. [Weg99]).

Das Ziel der folgenden Definitionen ist es, unter den Problemen in $\mathcal{NP}$ die in einer gewissen Weise schwierigsten Probleme zu charakterisieren und dann zu zeigen, dass diese vermutlich nicht in $\mathcal{P}$ liegen. Dies ergibt dann eine effektive Möglichkeit, auch für andere (Nicht-Entscheidungs-)Probleme zu zeigen, dass sie vermutlich nicht in $\mathcal{P}$ liegen, also nicht effizient lösbar sind.

Dazu ist es nötig, die Schwierigkeit verschiedener Entscheidungsprobleme vergleichen zu können:

Definition 1.3.14
Seien zwei Sprachen L_1 und L_2 über Alphabeten Σ_1 und Σ_2 gegeben. Dann heißt L_1 ***polynomiell reduzierbar*** *auf L_2 ($L_1 \leq_p L_2$), falls es eine in polynomieller Zeit berechenbare Transformationsfunktion $f : {\Sigma_1}^* \to {\Sigma_2}^*$ gibt, so dass für alle $x \in {\Sigma_1}^*$ gilt:*

$$x \in L_1 \qquad \Leftrightarrow \qquad f(x) \in L_2.$$

Solche Reduktionen haben zwei wichtige Eigenschaften: Sie sind transitiv und falls L_2 effizient lösbar ist, so folgt aus $L_1 \leq_p L_2$, dass auch L_1 effizient lösbar ist.

Mit diesem Begriff können nun die schwierigen Probleme in $\mathcal{NP}$ charakterisiert werden:

Definition 1.3.15

- *Eine Sprache L heißt $\mathcal{NP}$-hart, falls alle Probleme aus $\mathcal{NP}$ polynomiell reduzierbar auf L sind, d.h. $L' \leq_p L \ \ \forall L' \in \mathcal{NP}$.*
- *Eine $\mathcal{NP}$-harte Sprache $L \in \mathcal{NP}$ heißt $\mathcal{NP}$-vollständig.*

Das folgende Lemma liefert das wesentliche Mittel, um für ein Entscheidungsproblem nachzuweisen, dass es $\mathcal{NP}$-hart ist:

Lemma 1.3.16
Ist L_1 $\mathcal{NP}$-hart und $L_1 \leq_p L_2$, so ist auch L_2 $\mathcal{NP}$-hart.
Beweis:
Folgt direkt aus der Definition und der Transitivität von $\leq_p$. ■

Nach dem Satz von Cook ist das sogenannte Satisfiability-Problem, kurz SAT (vgl. [Weg99]), $\mathcal{NP}$-vollständig. Ein ebenfalls $\mathcal{NP}$-vollständiger Spezialfall davon ist das Problem 3-SAT:

Problem 1.3.17 (3-SAT)
Seien n Variablen $x_1, \ldots, x_n$ und m Klauseln

$$c_1, \ldots, c_m \in \{l_1 \vee l_2 \vee l_3 \mid l_i \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}\}$$

mit je drei Literalen gegeben.
*Dann besteht das Problem **3-SAT** darin, zu entscheiden, ob es eine Belegung für die Variablen $x_1, \ldots, x_n$ gibt, die alle Klauseln $c_1, \ldots, c_m$ erfüllt.*

Satz 1.3.18
3-SAT ist $\mathcal{NP}$-vollständig.

Ausgehend von SAT und 3-SAT wurde für viele weitere, wichtige Probleme gezeigt, dass sie $\mathcal{NP}$-vollständig sind. Viele dieser Probleme sind in [GJ79] zusammengefasst.

Die entscheidende Bedeutung des Studiums $\mathcal{NP}$-vollständiger Probleme für die Frage nach der Existenz effizienter Algorithmen zeigt der folgende Satz:

Satz 1.3.19
Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt:

- *$L \in \mathcal{P} \ \Rightarrow \ \mathcal{P} = \mathcal{NP}$ $\quad$ ($\Rightarrow L' \in \mathcal{P}$ für alle $\mathcal{NP}$-vollständigen Probleme L'),*
- *$L \notin \mathcal{P} \ \Rightarrow \ L' \notin \mathcal{P}$ für alle $\mathcal{NP}$-vollständigen Probleme L' (und damit $\mathcal{P} \neq \mathcal{NP}$).*

Entweder gibt es also für alle $\mathcal{NP}$-vollständigen Probleme effiziente Algorithmen oder kein einziges $\mathcal{NP}$-vollständiges Problem ist effizient lösbar. Da es sehr viele (darunter auch sehr wichtige) $\mathcal{NP}$-vollständige Probleme gibt, aber für alle diese Probleme noch immer kein effizienter Algorithmus gefunden werden konnte, gilt folgende Vermutung als sehr wahrscheinlich.

Vermutung 1.3.20
Es gilt $\mathcal{P} \neq \mathcal{NP}$, d.h. $\mathcal{NP}$-harte Probleme sind nicht effizient lösbar.

Unter dieser Annahme kann also in gewisser Weise gezeigt werden, dass ein Problem nicht effizient lösbar, d.h. schwierig ist.

Aus der $\mathcal{NP}$-Vollständigkeitstheorie, die sich nur auf Entscheidungsprobleme bezieht, kann auch die vermutliche Ineffizienz anderer (Nicht-Entscheidungs-)Probleme abgeleitet werden, etwa von Suchproblemen:

Definition 1.3.21
*Ein **Suchproblem** Π wird beschrieben durch die Menge D_Π der gültigen Eingaben für Π und durch Mengen $S_\Pi(I)$ für alle $I \in D_\Pi$, die zu jeder Eingabe die gesuchten Lösungen beschreiben.*
Ein Algorithmus löst das Suchproblem Π, wenn er zu einer Eingabe $I \in D_\Pi$ ein Element aus $S_\Pi(I)$ (bzw. für $S_\Pi(I) = \emptyset$ die Antwort „Nein“) ausgibt.

Durch Suchprobleme lassen sich sehr viele in der Praxis vorkommende Arten von Problemen erfassen. Insbesondere lassen sich auch Entscheidungsprobleme L durch $S_L(x) =$ „Ja“ für $x \in L$ und $S_L(x) = \emptyset$ für $x \notin L$ als Suchprobleme auffassen.

Für solche Suchprobleme muss der Reduktionsbegriff aus Definition 1.3.14 ein wenig erweitert werden. Ein Problem Π wird als Turing-reduzierbar auf ein anderes Problem Π' bezeichnet, falls es eine sogenannte Orakelturingmaschine mit einem Orakel für Π' gibt, die Π löst. Eine Orakelturingmaschine mit einem Orakel für Π' ist eine deterministische Turingmaschine, die neben den normalen Operationen in einem Rechenschritt (sozusagen als Unterprogramm) zu einer Eingabe von Π' eine Lösung berechnen kann. Der Begriff der Turing-Reduzierbarkeit umfasst insbesondere auch den Begriff der polynomiellen Reduktion aus Definition 1.3.14.

Damit können nun auch $\mathcal{NP}$-harte Suchprobleme definiert werden:

Definition 1.3.22
Ein Suchproblem Π heißt $\mathcal{NP}$-hart, wenn es ein $\mathcal{NP}$-hartes Entscheidungsproblem L gibt, das auf Π Turing-reduzierbar ist.

Für $\mathcal{NP}$-harte Suchprobleme gilt analog wie schon bei den Entscheidungsproblemen, dass sie unter Annahme der Vermutung 1.3.20 nicht effizient lösbar sind.

Eine weitere recht weit verbreitete Annahme ist

Vermutung 1.3.23
Es gilt $\mathcal{NP} \neq$ co-$\mathcal{NP}$.

Diese Vermutung ist unter der oben beschriebenen Auffassung von $\mathcal{NP}$ als Klasse der Probleme, für die durch glückliches Raten die Antwort „Ja“ effizient verifiziert werden kann, recht plausibel:
Denn etwa für das Beispielproblem „Gibt es eine Lösung des Systems $\mathcal{S}$?“ könnte als Beweis für die Antwort „Ja“ einfach eine Lösung angegeben werden. Was aber als Beweis für die Antwort „Nein“, also für die Nichtexistenz einer Lösung, angegeben werden sollte, ist nicht ohne weiteres klar. Es ist also gut möglich, dass dieses Problem zwar in $\mathcal{NP}$, aber nicht in co-$\mathcal{NP}$ liegt.

Unter dieser Annahme kann nachgewiesen werden, dass ein Problem nicht $\mathcal{NP}$-hart ist:

Satz 1.3.24
Falls $\mathcal{NP} \neq$ co-$\mathcal{NP}$ gilt und $L \in \mathcal{NP} \cap$ co-$\mathcal{NP}$ ist, dann ist L nicht $\mathcal{NP}$-hart.

Neben $\mathcal{P}$, $\mathcal{NP}$ und co-$\mathcal{NP}$ werden in der Komplexitätstheorie noch viele weitere Komplexitätsklassen von Problemen betrachtet. Von diesen Klassen seien hier nur kurz die folgenden Klassen erwähnt:

- $\mathcal{PSPACE}$ bzw. $\mathcal{EXPSPACE}$, die die Probleme umfassen, die mit polynomiell bzw. exponentiell viel Speicherplatz auskommen und
- $\mathcal{EXPTIME}$ bzw. $\mathcal{EXPEXPTIME}$, die die Probleme umfassen, die in exponentieller bzw. doppelt exponentieller Zeit lösbar sind.

Für diese Problemklassen ist folgende Hierarchie bekannt:

$$\mathcal{P} \subseteq \mathcal{NP} \cap \text{co-}\mathcal{NP} \subseteq \left\{ \begin{array}{c} \mathcal{NP} \\ \text{co-}\mathcal{NP} \end{array} \right\} \subseteq \mathcal{NP} \cup \text{co-}\mathcal{NP}$$
$$\subseteq \mathcal{PSPACE} \subseteq \mathcal{EXPTIME} \subseteq \mathcal{EXPSPACE} \subseteq \mathcal{EXPEXPTIME}.$$

Auch der Vollständigkeitsbegriff kann auf diese Klassen (allerdings bezüglich anderer Reduktionen) verallgemeinert werden. Dabei kann ein $\mathcal{C}$-vollständiges Problem immer als in der Klasse $\mathcal{C}$ „möglichst weit außen“ liegendes Problem aufgefasst werden, also insbesondere als ein Problem, dass nicht in der nächstniedrigeren Klasse $\mathcal{C}'$ liegt, falls sich $\mathcal{C}$ und $\mathcal{C}'$ unterscheiden.

Zum Abschluß noch eine wichtige Bemerkung zur Einordnung der Komplexitätstheorie in der Kryptographie:

Bemerkung 1.3.25

Der hier beschriebene Effizienzbegriff aus der Komplexitätstheorie erfasst ausschließlich worst case Rechenzeiten. Ein Problem heißt also schon dann nicht effizient lösbar, wenn es wenige Instanzen dieses Problems gibt, die schwierig zu lösen sind. Es ist aber durchaus möglich, dass ein Problem, das im worst case exponentielle Rechenzeit benötigt, in vielen Fällen, möglicherweise sogar im Durchschnitt in polynomieller Zeit lösbar ist.
In der Kryptographie ist aber eher die durchschnittliche Schwierigkeit von Interesse. Beispielsweise sollte das Problem, einen Chiffretext ohne Kenntnis des geheimen Schlüssels zu entschlüsseln, in möglichst allen Fällen schwierig zu lösen sein. Wäre nur der worst case als schwierig bekannt (das Problem also beispielsweise $\mathcal{NP}$-hart), könnten durchaus viele Chiffretexte leicht zu entschlüsseln sein.

Kapitel 2

HFE — Hidden Field Equations

Das Kryptosystem HFE basiert auf dem von Hideki Imai und Tsutomu Matsumoto in [IM88] vorgestellten System C^*, das Jacques Patarin in [Pat95] auf der Eurocrypt 1995 gebrochen hat. Im darauffolgenden Jahr hat Patarin dann selber in [Pat96a] als Reparaturvorschlag für C^* unter anderem das Verfahren HFE vorgestellt.
HFE steht für „**H**idden **F**ield **E**quations“, was andeuten soll, dass die Umkehrung der bei diesem Kryptosystem verwendeten öffentlichen Funktion sozusagen auf in einem Erweiterungskörper versteckten Gleichungen basiert.

In Abschnitt 2.1 werden die grundlegenden Verfahren C^* und HFE vorgestellt. Dort wird zunächst der beiden Verfahren gemeinsame Aufbau beschrieben, um dann darzustellen, worin sich die beiden Verfahren unterscheiden. Da die Veränderungen, die von C^* zu HFE führen, unter anderem bewirken, dass die bei HFE verwendete Einwegfunktion nicht mehr bijektiv ist, wird im zweiten Abschnitt dieses Kapitels beschrieben, was beachtet werden muss, damit HFE trotzdem in der Praxis angewendet werden kann.
Anschließend werden in Abschnitt 2.3 einige Eigenschaften von HFE betrachtet und beschrieben, welche Auswirkungen diese auf die Wahl der Parameter haben. Dabei wird besonderes Augenmerk auf die fehlende Bijektivität der Einwegfunktion gelegt und festgestellt, wie groß die in Abschnitt 2.2 beschriebenen Veränderungen sein müssen, damit das Entschlüsseln und Signieren mit HFE nur mit zu vernachlässigender Wahrscheinlichkeit misslingt.
Im vierten Abschnitt werden sogenannte Perturbationen für HFE vorgestellt. Dies sind kleine Variationen, die das Verfahren leicht modifizieren und Angriffe schwieriger machen sollen. In Abschnitt 2.5 werden dann noch einige grundlegende Sicherheitsaspekte von HFE vor allem aus komplexitätstheoretischer Sicht beschrieben.

2.1 Das HFE-Schema

In diesem Abschnitt wird die grundlegende Struktur von HFE beschrieben. Da HFE als Verallgemeinerung von C^* entstanden ist, wird zunächst das allgemeine Verfahren, das beiden Systemen zugrunde liegt, vorgestellt. Dann wird kurz C^* erklärt und die Attacke aus [Pat95] skizziert, mit der Patarin dieses Verfahren gebrochen hat. Anschließend wird eine der Möglichkeiten beschrieben, die Patarin vorgeschlagen hat, um diese Attacke zu verhindern, nämlich das HFE-System.

C^* und HFE arbeiten über endlichen Körpern. Soweit nichts anderes definiert wird, sei dazu in der gesamten Arbeit — wie schon in Abschnitt 1.3.1 — $K = \mathbb{F}_q$ ein endlicher Körper der Charakteristik p und $L = \mathbb{F}_{q^n}$ ein Erweiterungskörper von K vom Grad n. Dies alles sind öffentliche Parameter der beiden Kryptosysteme.

2.1.1 Das zugrunde liegende Schema

Grundlage für C^* und HFE ist — wie eigentlich für alle bekannten Public-Key-Kryptosysteme — eine Einwegfunktion mit Hintertür. Bei diesen Verfahren wird eine Funktion

$$\begin{array}{rccc} P: & K^n & \to & K^n \\ & (x_1, \ldots, x_n) & \mapsto & (p_1(x_1, \ldots, x_n), \ldots, p_n(x_1, \ldots, x_n)) \end{array}$$

verwendet, die durch n quadratische Polynome $p_1, \ldots, p_n$ in den n Variablen $x_1, \ldots, x_n$ dargestellt werden kann, d.h. in den in Abschnitt 1.3.1 eingeführten Begriffen gilt

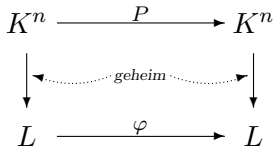
$$M(P) \in \mathcal{M}_{n,2}.$$

Offensichtlich ist $P(x)$ für ein gegebenes $x \in K^n$ effizient berechenbar. Die umgekehrte Richtung, zu einem gegebenen $y \in K^n$ ein $x \in K^n$ mit $P(x) = y$ zu finden, oder anders ausgedrückt zu $(y_1, \ldots, y_n) \in K^n$ das quadratische Gleichungssystem

$$\begin{array}{rcl} p_1(x_1, \ldots, x_n) - y_1 & = & 0 \\ \vdots \qquad\qquad\quad \vdots & & \vdots \\ p_n(x_1, \ldots, x_n) - y_n & = & 0 \end{array}$$

zu lösen, ist ein spezieller Fall des als schwierig geltenden Problems MQ, das in Abschnitt 2.5 definiert wird. Die Schwierigkeit dieses Problems ist eine wichtige Grundlage für die Sicherheit von HFE.
Ob die Funktion P wirklich eine Einwegfunktion im strengen Sinne darstellt, also nicht effizient umkehrbar ist, ist nicht bekannt. Da aber die Sicherheit von Kryptosystemen, die auf diese Weise aufgebaut sind, immer auf der Annahme beruht, dass die zugrunde liegende Funktion eine Einwegfunktion ist, wird im Folgenden im Zusammenhang mit der Funktion P oft von einer Einwegfunktion gesprochen.

Die Hintertür in HFE und C^* basiert auf der Idee, in der multivariaten Funktion $P : K^n \to K^n$ über dem kleineren Körper K eine in gewisser Weise einfachere, univariate Funktion $\varphi : L \to L$ über dem größeren Körper L zu verstecken, d.h. die Verfahren arbeiten nach folgendem Schema:

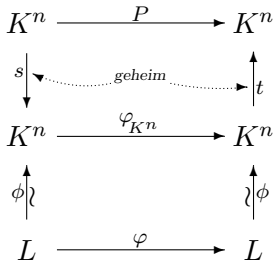


Dabei muss natürlich der Zusammenhang zwischen K^n und L geheim bleiben und die Funktion φ muss **effizient umkehrbar** sein, d.h. für alle $b \in Bild(\varphi)$ muss sich effizient ein $a \in L$ mit $\varphi(a) = b$ finden lassen. Das spezielle Aussehen der zu versteckenden Funktion φ im Falle von C^* bzw. HFE wird weiter unten ausführlich beschrieben, da sich die Verfahren C^* und HFE im Wesentlichen in der Wahl von φ unterscheiden.

Im Folgenden wird beschrieben, wie φ so in P versteckt werden kann, dass P ohne die Kenntnis des Zusammenhangs von P und φ vermutlich nicht effizient umkehrbar ist:

Sei dazu $\varphi : L \to L$ mit $M(\varphi) \in \mathcal{M}_{n,2}$. Dann könnte nach Wahl eines Isomorphismus $\phi : L \to K^n$ eigentlich direkt $\varphi_{K^n} := \phi \circ \varphi \circ \phi^{-1} : K^n \to K^n$ (vgl. Abschnitt 1.3.1) als öffentliche Funktion P gewählt werden. Allerdings hinge dann die Sicherheit des Verfahrens, nämlich das Verstecken der Funktion φ, direkt von der Geheimhaltung des Isomorphismus ϕ ab.

Um die Sicherheit nicht darauf stützen zu müssen, wird φ_{K^n} noch durch zwei geheime, bijektive, affine Abbildungen $s, t : K^n \to K^n$ versteckt. Dabei bedeutet affin, dass s und t Kompositionen einer linearen Abbildung und einer Translation sind.
Als öffentliche Funktion wird dann $P := t \circ \varphi_{K^n} \circ s = t \circ \phi \circ \varphi \circ \phi^{-1} \circ s$ gewählt:



Diese so entstandene, öffentliche Funktion P hat offensichtlich die folgenden von der Wahl von φ abhängigen Eigenschaften:

- Ist φ effizient umkehrbar und sind ϕ, s und t bekannt, so ist auch P effizient umkehrbar.

- Da s und t bijektiv gewählt sind, ist P genau dann bijektiv, wenn dies auch für φ gilt.

- Da s und t affin sind, gilt für alle $k \in \mathbb{N}$:

$$M(P) \in \mathcal{M}_{n,k} \quad \Leftrightarrow \quad M(\varphi) = M\left(\varphi_{K^n}\right) \in \mathcal{M}_{n,k} \qquad (\Leftrightarrow U(\varphi) \in \mathcal{U}_{n,k})$$

2.1.2 Das Verfahren C^*

In diesem Abschnitt wird beschrieben, wie die Funktion φ im Falle von C^* gewählt wird.

An dieser Stelle muss zunächst angemerkt werden, dass das von Imai und Matsumoto in [IM88] vorgestellte Verfahren C^* eigentlich sogenannte Äste verwendet. Das bedeutet, dass die Funktion φ_{K^n} nicht durch ein einziges φ über einem einzelnen Oberkörper L vom Grad n bestimmt wird. Stattdessen wird n in $n = n_1 + \ldots + n_l$ zerlegt und die Funktion φ_{K^n} wird zusammengesetzt aus l Funktionen φ_i, die über Oberkörpern L_i vom Grad n_i gewählt werden. Patarin zeigt aber in [Pat95] und [GP97a], dass sich diese Zerlegung von einem Angreifer herausfinden lässt, und dass die Kryptoanalyse, die sich dann auf die einzelnen Äste beschränken kann, dadurch höchstens einfacher wird. Da solche Äste aus diesem Grund in HFE keine Anwendung finden, wird C^* hier nur in der einfachen Form mit einem einzelnen Ast — also wie oben in der allgemeinen Form beschrieben — betrachtet.
Eine weitere Analyse der Bestimmung der Äste kann in [Fel01] nachgelesen werden.

Als versteckte Funktion φ wird für C^* eine monomiale Abbildung der Form

$$\varphi: \begin{array}{ccc} L & \to & L \\ a & \mapsto & a^{1+q^\theta} \end{array}$$

gewählt mit einem $\theta \in \{1, \ldots, n-1\}$, so dass $1 + q^\theta$ und $q^n - 1$ teilerfremd sind. Dies führt gleich zu der Feststellung, dass C^* nur über Körpern der Charakteristik $p = 2$ durchführt werden kann, da für ungerades q sowohl $1 + q^\theta$ als auch $q^n - 1$ gerade sind. Ansonsten hat diese Wahl von φ aber durchaus einige Vorteile:

- Da $1 + q^\theta$ und $q^n - 1$ teilerfremd sind, existiert ein $h \in \{1, \ldots, q^n - 1\}$ mit

$$\left(1 + q^\theta\right) \cdot h \equiv 1 \mod q^n - 1.$$

  Damit gilt für alle $a \in L$

$$(\varphi(a))^h = a^{\left(1+q^\theta\right)\cdot h} = a.$$

  Somit ist φ *bijektiv*. Wie oben gesehen, ist damit auch die aus diesem φ resultierende öffentliche Funktion P bijektiv, was für die Anwendung des Systems zur Folge hat, dass zu jedem Chiffretext genau ein Klartext bzw. zu jeder Nachricht genau eine Signatur existiert.

- Zudem ist $U(\varphi) \in \mathcal{U}_{n,2}$ und damit nach Satz 1.3.9 auch $M(\varphi) \in \mathcal{M}_{n,2}$. Da sich — wie oben beschrieben — diese Eigenschaft auf P überträgt, ist also wie gewünscht auch $M(P) \in \mathcal{M}_{n,2}$, d.h. P kann durch quadratische Polynome beschrieben werden.

Leider hat diese Wahl von φ aber den Nachteil, dass die Umkehrung von P auch ohne Kenntnis von s und t effizient gelingt, ein auf diesem P aufbauendes Kryptoverfahren also gebrochen werden kann, wie Patarin in [Pat95] zeigt. Ein so gewähltes P ist also leider keine Einwegfunktion.
Die in [Pat95] beschriebene Attacke basiert auf der Feststellung, dass zwischen dem Klartext $(x_1, \ldots, x_n)$ und der verschlüsselten Nachricht $(y_1, \ldots, y_n) = P(x_1, \ldots, x_n)$ immer Relationen der Form

$$\sum\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \tag{2.1}$$

existieren, die Grad 1 in den x_i und Grad 1 in den y_i haben. Das liegt daran, dass für $a = \phi^{-1}(s(x_1, \ldots, x_n))$ und $b = \phi^{-1}\left(t^{-1}(y_1, \ldots, y_n)\right)$, also $b = \varphi(a) = a^{1+q^\theta}$ auch immer (durch Potenzieren mit $q^\theta - 1$ und Multiplikation mit $a \cdot b$)

$$a \cdot b^{q^\theta} = a^{q^{2\theta}} \cdot b \tag{2.2}$$

gilt. Da aber $a \mapsto a^{q^{2\theta}}$ und $b \mapsto b^{q^\theta}$ K-linear und s und t affin sind, liefert (2.2) bezüglich einer Basis betrachtet gerade n Relationen der Form (2.1).
Solche Relationen lassen sich auch für einen Angreifer, der s und t nicht kennt, leicht herausfinden, indem er durch Anwendung von P genügend viele Paare $(x, y) = (x, P(x))$ erzeugt, diese in (2.1) einsetzt und das sich daraus für die γ_{ij}, α_i, β_i und δ_0 ergebende lineare Gleichungssystem löst. Mit Hilfe solcher Relationen, die linear in den x_i sind, erhält der Angreifer dann zu einem gegebenen $(y_1, \ldots, y_n)$ durch Einsetzen in die Relationen ein lineares Gleichungssystem in den x_i, das ihm zumindest einen großen Teil der Werte der x_i liefert (abhängig davon, wie viele linear unabhängige Relationen der Form (2.1) gefunden werden können).
Eine ausführlichere Beschreibung und Analyse dieser Attacke wird in [Fel01] durchgeführt. Eine Verallgemeinerung dieses Vorgehens, die sogenannte Attacke über implizite Gleichungen, wird in Kapitel 5 beschrieben.

Eine Möglichkeit, diese Attacke zu verhindern, ist, für $U(\varphi)$ statt eines Monoms a^{1+q^θ}, ein mehr oder weniger beliebiges Polynom $U(\varphi) =: f \in \mathcal{U}_{n,2}$ zu verwenden, so dass Relationen der Form (2.1) (hoffentlich) nicht mehr existieren. Solche Polynome verwendet das Verfahren HFE, das im folgenden Abschnitt vorgestellt wird.
Andere Möglichkeiten, C^* zu reparieren, hat Patarin zusammen mit Louis Goubin beispielsweise in [Pat96a],[Pat96b],[GP97b],[GP97a] und [CGP98b] vorgeschlagen.

2.1.3 Das Verfahren HFE

Bei HFE wird die versteckte Funktion $\varphi : L \to L$ durch ein Polynom der Form

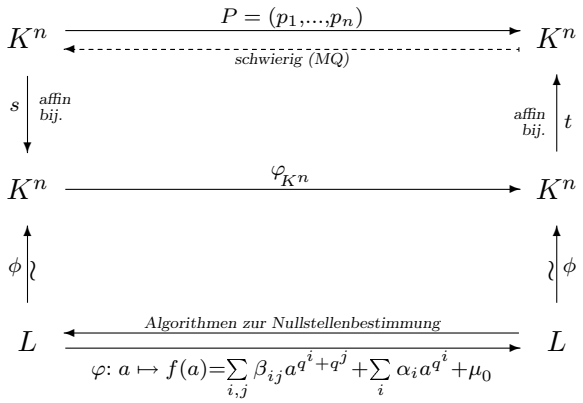
$$f(x) = \sum_{i,j=0}^{n-1} \beta_{ij} x^{q^i+q^j} + \sum_{i=0}^{n-1} \alpha_i x^{q^i} + \mu_0 \in L[x]$$

mit β_{ij}, α_i, $\mu_0 \in L$ als $\varphi : a \mapsto f(a)$ dargestellt. Für den Fall $q = 2$ sollte noch zusätzlich $\beta_{n-1,n-1} = 0$ gefordert werden, damit immer $f = U(\varphi)$ gilt.
Ein solches Polynom f wird im Folgenden auch als **HFE-Polynom** und sein Grad mit $d := grad(f)$ bezeichnet.
HFE-Polynome sind also genau die durch Gewicht 2 beschränkten, univariaten Darstellungen $f \in \mathcal{U}_{n,2}$, d.h. es gilt auch hier $M(P) \in \mathcal{M}_{n,2}$.

Leider geht bei HFE allerdings die Bijektivität der Abbildung φ und damit natürlich auch die von P verloren, da keine einfache Möglichkeit bekannt ist, bijektive Abbildungen φ mit $U(\varphi) \in \mathcal{U}_{n,2}$ zu finden. Dies hat zur Folge, dass die Entschlüsselung von Nachrichten nicht immer eindeutig möglich ist, und dass nicht zu jeder Nachricht eine Signatur existiert. Um diese Probleme zu vermeiden, müssen beim Verschlüsseln und Signieren gewisse Redundanzen bzw. Randomisierungen eingebaut werden, was in Abschnitt 2.2 genauer beschrieben wird. In der Praxis ist dies allerdings meist kein großes Problem, da es zwar theoretisch bis zu d Lösungen x für $P(x) = y$ geben kann, es aber häufig nur 0 bis 2 Lösungen gibt, wie in Abschnitt 2.3.3 gezeigt wird.

Trotz der fehlenden Bijektivität ist die Abbildung φ dennoch effizient umkehrbar, solange der Grad d nicht zu groß ist. Denn die Problemstellung, zu einem gegebenen $b \in L$ diejenigen $a \in L$ zu finden, die $\varphi(a) = b$ erfüllen, entspricht gerade der Aufgabe, die Nullstellen von $f(x) - b \in L[x]$ zu bestimmen. Dafür sind aber viele Algorithmen bekannt, die in d und n polynomielle Laufzeiten haben. Diese werden in Abschnitt 2.3.2 kurz vorgestellt.

Das folgende Diagramm veranschaulicht noch einmal den Aufbau des HFE-Kryptosystems:



In den folgenden Abschnitten sei, sofern nicht ausdrücklich etwas anderes angegeben wird, mit f immer ein HFE-Polynom bezeichnet, mit φ die entsprechende geheime Abbildung und mit P die dazugehörige, öffentliche Einwegfunktion.

2.2 Anwendung von HFE

In diesem Abschnitt wird beschrieben, wie HFE als Kryptosystem praktisch angewendet werden kann. Da der Hauptbestandteil von HFE, die Einwegfunktion P, schon im letzten Abschnitt beschrieben wurde, liegt der Schwerpunkt der Beschreibung in diesem Abschnitt darauf, welche Parameter veröffentlicht bzw. geheimgehalten werden müssen und wie das Problem der fehlenden Bijektivität von P gelöst werden kann.

Beim HFE-Kryptosystem müssen folgende Parameter veröffentlicht werden:

- Der Grundkörper K und damit natürlich auch implizit die Anzahl q seiner Elemente und seine Charakteristik p.
- Die Art und Weise, wie die Redundanz bzw. die Randomisierung verwendet werden, um das Problem der fehlenden Bijektivität zu umgehen. (Dies wird weiter unten in diesem Abschnitt näher erläutert.)
- Die Einwegfunktion P, gegeben durch die multivariate Darstellung $M(P)$ bestehend aus n quadratischen Polynomen $p_1, \ldots, p_n$ über K in n Variablen.

Dabei sind K, p, q, n und auch die Redundanz/Randomisierung sozusagen feste Parameter eines auf HFE aufbauenden Verfahrens, während jeder Benutzer einen eigenen öffentlichen Schlüssel P veröffentlichen kann. Solche öffentlichen Schlüssel sind bei HFE mit ca. $\frac{n^3}{2} \log_2 q$ Bits allerdings deutlich größer als die Public Keys, die bei RSA oder bei auf elliptischen Kurven aufbauenden Systemen verwendet werden.

Unbedingt geheimgehalten werden müssen nur die beiden bijektiven, affinen Transformationen s und t, die jeder Benutzer als geheimen Schlüssel auswählt.

Zu klären bleibt die Frage der Veröffentlichung noch für den Oberkörper L bzw. seinen durch den Isomorphismus ϕ gegebenen Zusammenhang zum K^n und für die versteckte Funktion φ :

- Der Oberkörper L und vor allem seine Interpretation als K^n durch den Isomorphismus ϕ sind für einen Benutzer des Kryptosystems, der Nachrichten verschlüsseln oder Signaturen verifizieren möchte, nicht von Bedeutung, sie müssen daher nicht veröffentlicht werden. Allerdings können sie auch genauso gut als feste, öffentliche Parameter behandelt werden. Denn, da n bekannt ist, ist — bis auf Isomorphie —

auch L bekannt, und anstatt einen anderen Isomorphismus ϕ zu verwenden, kann der Besitzer des geheimen Schlüssels das gleiche Ergebnis erhalten, indem er die geheimen Transformationen s und t entsprechend ändert.

- Auch die Kenntnis der versteckten Funktion φ ist für einen Benutzer zum Verschlüsseln oder Verifizieren natürlich nicht nötig, d.h. auch φ kann geheimgehalten werden. Dies wird von Patarin in [Pat96a] auch empfohlen, da es seiner Meinung nach die Sicherheit höchstens steigert.
  Zudem wird dort gezeigt, dass das Finden des geheimen Schlüssels (s, t) mit Hilfe der Kenntnis von φ gerade dem Problem „Isomorphism of Polynomials“ (IP) entspricht, das in Abschnitt 2.5.1 genauer beschrieben wird. Dieses Problem ist vermutlich nicht in polynomieller Zeit lösbar (siehe auch Abschnitt 2.5.2), weshalb es theoretisch wohl möglich wäre, die Funktion φ zu veröffentlichen. Allerdings müsste aufgrund bekannter, subexponentieller Algorithmen für IP der Parameter n größer gewählt werden als es sonst nötig ist.
  Der Vorteil einer Verwendung von veröffentlichten Funktionen φ für HFE läge darin, dass nicht jeder Benutzer selbst entscheiden muss, ob das φ, das er verwendet, sicher ist. Da die Komplexität mancher Attacken (vgl. Kapitel 5) stark von der versteckten Funktion φ abhängt, ist es nämlich wichtig, φ sorgfältig zu wählen. Die Verwendung von öffentlichen Funktionen φ gäbe dann einer Gruppe von Benutzern die Möglichkeit, zusammen nach einem sicheren φ zu suchen und dieses durch verschiedene, geheime Schlüssel für unterschiedliche, öffentliche Schlüssel zu verwenden.

Kryptographieverfahren dienen im Wesentlichen drei Zwecken: Verschlüsselung, Signatur und Authentifikation. Dies ist alles auch mit HFE möglich, wie im Folgenden beschrieben wird:

2.2.1 Verschlüsselung

Wie in Abschnitt 2.1 beschrieben, kann mit Hilfe des HFE-Schemas immer ein Klartext $x \in K^n$ zu einem Chiffretext $y = P(x) \in K^n$ verschlüsselt werden. Der naive Ansatz, eine Nachricht M zu verschlüsseln, wäre also, $x = M$ zu wählen (für eine geeignete Codierung von M in K^n) und dieses x zu verschlüsseln. Da aber die Verschlüsselungsfunktion P — wie oben bemerkt — nicht bijektiv und vor allem nicht injektiv sein muss, ist es möglich, dass zu einem $y = P(x)$ verschiedene $x \in K^n$ mit $P(x) = y$ existieren, d.h. y könnte im Allgemeinen nicht eindeutig entschlüsselt werden. Deshalb müssen zusätzlich zur (Verschlüsselung der) Nachricht M noch weitere von M abhängige, sozusagen redundante Informationen verschickt werden, die es dem Empfänger ermöglichen, das richtige Urbild x von den „falschen“ Urbildern unterscheiden zu können.

Dafür gibt es zwei verschiedene Ansätze, die sogenannte **innere Redundanz** und die **äußere Redundanz**.

Innere Redundanz bedeutet, dass schon *innerhalb* des Klartextes x zusätzlich zur Nachricht M die redundanten Informationen eingefügt werden.
Eine Möglichkeit dazu ist, als Klartext

$$x = M \parallel h(M)$$

zu wählen, wobei $h(M)$ beispielsweise die ersten Bits eines für M berechneten Hashwertes bezeichnet. Eine andere, elegante Art, die Redundanz direkt in x einzufügen, ist, M mit Hilfe von fehlererkennenden Codes zu codieren und als x das zugehörige Codewort zu wählen.
Wie groß dabei die einzufügende Redundanz sein sollte, d.h. wie viele Bits der Hashfunktion angehängt werden sollten bzw. wie groß die Informationsrate des Codes gewählt werden sollte, wird in Abschnitt 2.3.3 näher untersucht.

Ein Nachteil dieser Verfahren mit innerer Redundanz ist, dass die Menge der Klartexte, die gesendet werden können, eingeschränkt wird. Deshalb muss für diese Verfahren der Parameter n größer gewählt werden, um beispielsweise eine vollständige Suche auf allen Klartexten, die von der vereinbarten Form sind, unmöglich zu machen.

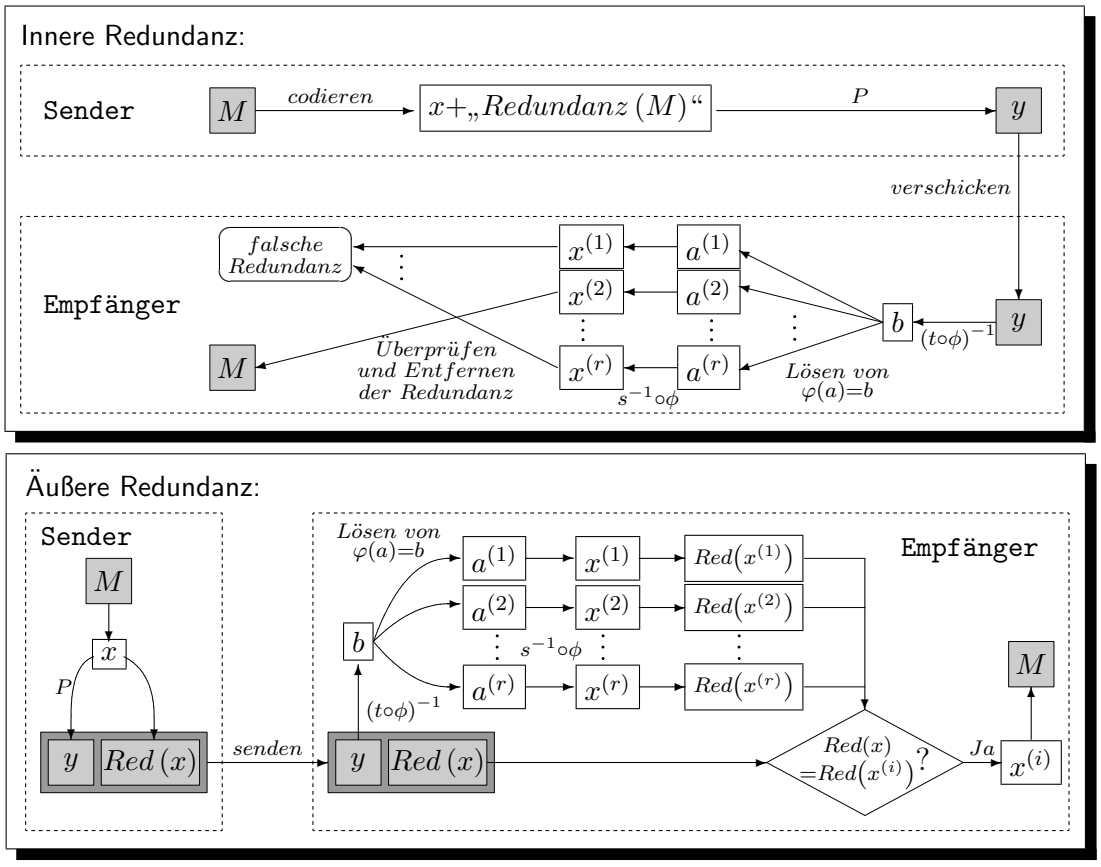
Äußere Redundanz bedeutet, dass direkt $x = M$ verwendet wird und dann zusätzlich zur verschlüsselten Nachricht $y = P(x)$ noch redundante Informationen über x versendet werden.
Eine Möglichkeit, äußere Redundanz hinzuzufügen, ist, mit Hilfe einer kollisionsresistenten Hashfunktion h als Chiffretext $Y = P(x) \parallel h(x)$ zu verschicken.
Alternativ kann auch $\widetilde{y} := (y_1, \ldots, y_n, \widetilde{y}_{n+1}, \ldots, \widetilde{y}_{n+k})$ verschickt werden, wobei die Werte $\widetilde{y}_{n+1}, \ldots, \widetilde{y}_{n+k}$ durch zusätzliche, zufällig gewählte, öffentliche Polynome erhalten werden.

Eine Mischform von innerer und äußerer Redundanz ergibt sich, wenn dabei die zusätzlichen Polynome durch die geheimen Bijektionen s und t mit den zu φ_{K^n} gehörigen Polynomen vermischt werden, so dass die zusätzlichen Polynome nach außen hin (z.B. für einen Angreifer) nicht erkennbar sind. Dieses Vorgehen entspricht gerade der Perturbation „+“, die in Abschnitt 2.4.2 genauer beschrieben wird.

Die Entschlüsselung kann wie in Abschnitt 2.1 angedeutet erfolgen: Der rechtmäßige Empfänger (d.h. Besitzer des geheimen Schlüssels) erhält durch die Umkehrung der ihm bekannten, geheimen Transformationen s und t ein $b = \phi^{-1}\left(t^{-1}(y)\right) \in L$ und kann dann die Lösungen $a^{(1)}, \ldots, a^{(k)}$ von $\varphi(a) = b$ bestimmen. Daraus erhält er die möglichen Nachrichten $x^{(i)} = s^{-1}\left(\phi\left(a^{(i)}\right)\right)$.

Wird ein Verfahren mit innerer Redundanz verwendet, kann er nun überprüfen, welches der $x^{(i)}$ bezüglich der Redundanzvorschrift korrekt aufgebaut ist, beispielsweise also ein Codewort des verwendeten fehlererkennenden Codes ist. Bei einem Verfahren mit äußerer Redundanz muss er überprüfen, welches der $x^{(i)}$ gerade die redundanten Informationen liefert, die er zusätzlich zum y empfangen hat.
Ist die eingebaute Redundanz genügend groß, so ist dabei die Wahrscheinlichkeit, dass das ursprüngliche x dadurch nicht identifiziert werden kann, vernachlässigbar klein (siehe auch Abschnitt 2.3.3). Dieses $x^{(i)}$ liefert dann (nach Entfernung der Redundanz bei einem Verfahren mit innerer Redundanz) die ursprüngliche Nachricht M.

Im Überblick ergibt dieses Vorgehen folgende Schemata:



Die Frage, wie viele redundante Informationen eingefügt werden müssen, damit (fast) immer eine eindeutige Entschlüsselung möglich ist, wird in Abschnitt 2.3.3 beantwortet, in dem die fehlende Bijektivität von P näher untersucht wird.

2.2.2 Signatur

Eine Eigenschaft von HFE ist, dass mit diesem System im Vergleich zu anderen gängigen Verfahren sehr kurze Signaturen erzeugt werden können. So gilt die beim auf HFE aufbauenden QUARTZ-Verfahren (siehe [CGP01]) verwendete Signatur von 128 Bit als die

nach der McEliece-Signatur (siehe [CFS01]) kürzeste, bislang nicht gebrochene Signatur (vergleiche [Cou01b]). Das McEliece-Schema ist allerdings im Gegensatz zu HFE deutlich ineffizienter in der Durchführung.

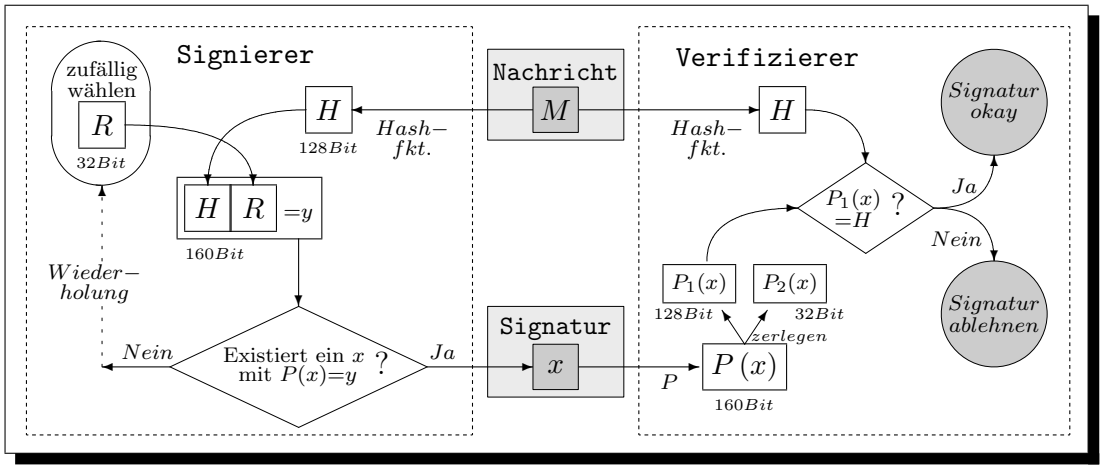
Das Problem beim Signieren mit HFE ist ebenfalls die fehlende Bijektivität von P, in diesem Fall genauer gesagt die fehlende Surjektivität. Denn dadurch existiert nicht zu jeder Nachricht $y = M \in K^n$ eine Signatur x mit $P(x) = y$. Der Ausweg ist nun, statt $y = M$ zu wählen, die Bildungsvorschrift für y so zu vereinbaren, dass y neben M auch noch einen frei wählbaren Anteil enthält. Ist dieser Anteil groß genug, gibt er dem Signierer genügend Freiheiten, um zu einem gegebenen M ein y zu finden, für das ein x mit $P(x) = y$ existiert.
Im Folgenden werden nun zwei verschiedene Beispiele von Verfahren zum Bilden solcher HFE-Signaturen vorgestellt, die Patarin in [Pat96a] vorschlägt.

160-Bit-Signatur

In Abschnitt 2.2.1 wurde als eine Möglichkeit zum Einfügen von Redundanz vorgeschlagen, zusätzliche öffentliche Polynome hinzuzufügen, die die redundanten Informationen liefern. Da die für das Signieren benötigte **Randomisierung**, also das Hinzufügen eines variablen, frei wählbaren Anteils, in gewisser Hinsicht gerade das Gegenteil dazu ist, ist es naheliegend, dass dies beispielsweise durch das Verbergen eines Teils der öffentlichen Polynome geschehen kann.
In dem hier vorgestellten Beispiel sei dafür $n = 160$, $q = p = 2$ und f ein dazu passendes HFE-Polynom. Zusammen mit den geheimen Transformationen s und t ergibt dies eine Funktion P mit einer multivariaten Darstellung $M(P) = (p_1, \ldots, p_{160})$, von der allerdings nur die ersten 128 Polynome $p_1, \ldots, p_{128}$ veröffentlicht werden. Die übrigen 32 Polynome bleiben geheim und dienen dazu, dem Signierer eine gewisse Freiheit bei der Wahl der zu signierenden Nachricht zu gewähren. Dieses Vorgehen ist sehr ähnlich der Perturbation „$-$“, die in Abschnitt 2.4.1 vorgestellt wird.

Um den 128-Bit-Hashwert H einer Nachricht zu signieren, geht der Signierer nun folgendermaßen vor: Er wählt $y = (y_1, \ldots, y_{160})$ mit $(y_1, \ldots, y_{128}) = H$, ergänzt mit einem beliebigen (zufälligen) 32-Bit-String $(y_{129}, \ldots, y_{160})$. Zu diesem y versucht er nun auf die bekannte Art und Weise ein x mit $P(x) = y$ zu finden. Existiert ein solches x, ist dies die gesuchte Signatur, da natürlich insbesondere $(p_1(x), \ldots, p_{128}(x)) = (y_1, \ldots, y_{128}) = H$ gilt. Ist kein solches x zu finden, wiederholt der Signierer einfach solange dieses Verfahren mit neuen, zufälligen 32-Bit-Strings $(y_{129}, \ldots, y_{160})$, bis er eine Signatur findet.
Dabei ist die Wahrscheinlichkeit, dass der Signierer schon nach wenigen Versuchen ein solches x findet, sehr hoch, da, wie in Abschnitt 2.3.3 gezeigt wird, vermutlich nur zu ca. jedem dritten y kein Urbild x existiert.

Um eine solche Signatur $S = x$ zu überprüfen, genügt es für einen Verifizierer, mit Hilfe der öffentlichen Polynome zu überprüfen, ob $(p_1(x), \ldots, p_{128}(x)) = H$ gilt. Zusammengefasst wird dieses Vorgehen in folgendem Schema:



Bei der 160-Bit-Signatur wird der zu signierende Hashwert H als fest vorgegeben angesehen, weshalb die Randomisierung „außen“ in zusätzlichen Stellen hinzugefügt werden muss. Dies führt dazu, dass die Signatur deutlich länger ist, als der zu signierende Wert. Dieser Effekt kann beispielsweise mit Hilfe der folgenden Konstruktion vermieden werden:

ca.128-Bit-Signatur

Um eine Signatur zu erreichen, die kaum länger als die Länge des Hashwertes ist, muss schon der Hashwert selber randomisiert werden. Dies kann beispielsweise folgendermaßen geschehen:
Sei M die Nachricht, die signiert werden soll, h eine öffentliche, kollisionsresistente Hashfunktion mit einer Ausgabelänge von 128 Bits und P die öffentliche Funktion eines HFE-Systems mit $n = 128$ und $q = p = 2$.
Der Signierer wählt dann eine beliebige, kleine, natürliche Zahl R, in deren Binärdarstellung der String „10000“ nicht vorkommt und berechnet daraus den zu signierenden Hashwert als

$$H \quad = \quad h(R \parallel 10000 \parallel M).$$

Existiert zu diesem $y = H$ ein x mit $P(x) = y$ (das wie gewohnt mit Hilfe des geheimen Schlüssels bestimmt werden kann), dann wird die Signatur S gebildet durch

$$S = R \parallel x.$$

Existiert ein solches x nicht, kann der Signierer den gesamten Vorgang solange mit einem neuen R wiederholen, bis er eine Signatur gefunden hat. Dabei ist es sinnvoll mit kleinen Werten für R zu beginnen, da die Länge der Signatur gerade $128 + \lceil \log_2 R \rceil$ Bits beträgt.

Der Verifizierer kann aus S aufgrund der festen Länge von x (128 Bits) zunächst R und x trennen und braucht dann nur noch zu verifizieren, dass

$$P(x) = h(R \parallel 10000 \parallel M)$$

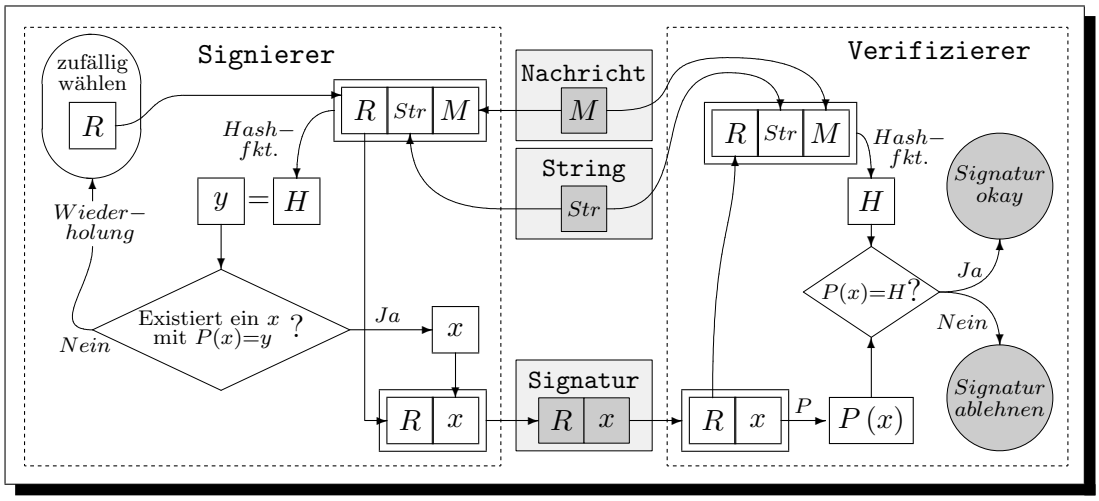
gilt.

Der String „10000“ ist nur ein Beispiel und kann durch einen beliebigen, anderen String Str ersetzt werden, der dazu geeignet ist, aus $R \parallel Str \parallel M$ eindeutig R und M zurückzugewinnen.
Ein String mit einer solcher Eigenschaft ist nötig, denn wäre

$$R \parallel Str \parallel M \quad = \quad R' \parallel Str \parallel M',$$

dann wäre zu einer Signatur $S = R \parallel x$ von M natürlich auch $S' = R' \parallel x$ eine gültige Signatur für M'. Dies würde einem Angreifer also ermöglichen, Signaturen für Nachrichten zu erhalten, die der rechtmäßige Besitzer des geheimen Schlüssels nie signiert hat, was natürlich vermieden werden sollte.

Einen Überblick über dieses Verfahren liefert das folgende Schema:



Es ist sogar möglich, auf HFE basierende Signaturschemata mit noch kürzeren Signaturen (z.B. etwa 64 Bit) zu bilden, wie Patarin sie beispielsweise in [Pat96a] beschreibt. Diese haben allerdings durch ihre verschachtelte Konstruktion den Nachteil, dass das Verifizieren und vor allem das Signieren deutlich aufwendiger werden.

2.2.3 Authentifikation

Natürlich kann auch HFE, wie jedes asymmetrische Kryptographieverfahren, mit dem verschlüsselt oder signiert werden kann, zur Authentifikation verwendet werden. Dazu sendet der Verifizierer als Herausforderung eine verschlüsselte bzw. zu signierende Nachricht an den angeblichen Geheimnisträger und erwartet als Beweis für die Kenntnis des geheimen Schlüssels den Klartext bzw. eine gültige Signatur.

2.3 Wahl der Parameter

In diesem Abschnitt werden folgende Aspekte von HFE-Systemen ein wenig näher betrachtet und daraus Schlüsse für die Wahl der Parameter gezogen:

- Welche Abbildungen $P : K^n \to K^n$ bzw. multivariate, quadratische Polynomsysteme $M(P)$ können auftreten ?
- Auf welche Weise und vor allem wie schnell kann eine Nullstelle von $f(x) - b$ bestimmt und damit zu y eine Lösung x mit $P(x) = y$ gefunden werden ?
- Wie viele verschiedene Lösungen von $P(x) = y$ sind zu erwarten, und was bedeutet dies für die einzufügende Redundanz bzw. Randomisierung?

2.3.1 Charakterisierung der auftretenden Polynomsysteme

In Satz 1.3.9 wurde gezeigt, dass Polynomsysteme der Form

$$(p_1, \ldots, p_n) \text{ mit } grad(p_i) \leq 2 \text{ und } grad_{x_j}(p_i) \leq q - 1$$

und Polynome der Form

$$f = \sum a_i x^i \in L[x] \text{ mit } grad(f) \leq q^n - 1 \text{ und } a_i = 0 \text{ für alle } i \text{ mit } G_q(i) > 2,$$

also HFE-Polynome, in eineindeutiger Beziehung zueinander stehen. Zu jedem beliebigen quadratischen Polynomsystem mit n Polynomen in n Variablen existiert also ein HFE-Polynom $f = U(\varphi)$ zu einer Abbildung $\varphi : L \to L$, aus der sich das Polynomsystem als $M(\varphi)$ erhalten lässt. Da auch die bijektiven, affinen Transformationen s und t die Menge aller quadratischen Polynomsysteme mit n Variablen und n Gleichungen bijektiv auf sich selber abbilden, kann also zumindest theoretisch in einem HFE-System jedes beliebige Polynomsystem $M(P) \in \mathcal{M}_{n,2}$ auftreten.
Allerdings ist bei dieser Feststellung zu beachten, dass der Grad der HFE-Polynome, die mit solchen allgemeinen Polynomsystemen korrespondieren, nur durch $2 \cdot q^{n-1}$ beschränkt ist. Genauer gesagt hat das zu einem zufällig aus $\mathcal{M}_{n,2}$ gewählten Polynomsystem passende HFE-Polynom mit Wahrscheinlichkeit $\frac{q^n-1}{q^n}$ den vollen Grad $2 \cdot q^{n-1}$. Um durch ein HFE-Polynom also wirklich ein zufälliges Polynomsystem zu erzeugen, muss im Allgemeinen $d = grad(f)$ maximal, also gleich $2 \cdot q^{n-1}$ gewählt werden, was für die Praxis aber keine sinnvolle Wahl ist, wie im folgenden Abschnitt 2.3.2 gezeigt wird.

Für die Sicherheit eines Public-Key-Kryptosystems ist es vorteilhaft, wenn die verwendete öffentliche Einwegfunktion P aus einer möglichst großen Menge von Funktionen ausgewählt werden kann und dabei vor allem möglichst wenige algebraische Eigenschaften besitzt, die eine Umkehrung (ohne Kenntnis des geheimen Schlüssels) erleichtern.

Wie gerade beschrieben, unterscheiden sich die bei HFE vorkommenden Polynomsysteme $M(P)$ von zufälligen Systemen aus $\mathcal{M}_{n,2}$ gerade dadurch, dass ihre univariate Darstellung nicht beliebig aus $\mathcal{U}_{n,2}$ sondern mit beschränktem Grad d gewählt wird. Um die Sicherheit eines HFE-Systems zu steigern, scheint es also sinnvoll zu sein, diesen Grad d möglichst groß zu wählen. Dementgegen steht — wie im folgenden Abschnitt zu sehen ist — leider die Erfordernis, dass φ effizient umkehrbar sein muss.

2.3.2 Effiziente Umkehrbarkeit der versteckten Funktion φ

In Abschnitt 2.1 wurde schon beschrieben, dass der Hauptteil der Umkehrung der Einwegfunktion P (unter Kenntnis der geheimen Transformationen s und t) in der Umkehrung der versteckten Funktion φ besteht, also in der Bestimmung von Nullstellen des univariaten Polynoms $f(x) - b$ für ein fest vorgegebenes $b \in L$. Für die Nullstellenbestimmung über endlichen Körpern gibt es eine Reihe effizienter Algorithmen, von denen im Folgenden einige zusammen mit ihren Laufzeiten kurz vorgestellt werden.

Die nachfolgende Tabelle listet die sechs in [Pat96a] vorgeschlagenen Algorithmen zur Nullstellenbestimmung zusammen mit ihren Laufzeiten auf. Die dabei angegebenen Laufzeiten sind erwartete Laufzeiten in Bezug auf folgende Parameter:
Gegeben sei ein Polynom f vom Grad d über dem Körper $L = \mathbb{F}_{q^n}$ mit $q = p^m$, wobei m so gewählt ist, dass eine Addition oder Multiplikation in $\mathbb{F}_q$ in konstanter Zeit durchgeführt werden kann (beispielsweise durch Speicherung in einer Additions- und Multiplikationstabelle). Die angegeben Laufzeiten entsprechen dann den erwarteten Anzahlen von Grundoperationen in $\mathbb{F}_q$. (Das für ein HFE-System gewählte q erfüllt diese Anforderungen üblicherweise.)
Beschreibungen der klassischen Algorithmen 1) bis 3) können in [BGMMVY92] und [LN83] nachgelesen werden, während die Varianten 4) und 5) in [Pat96a] kurz erklärt werden. Der asymptotisch schnellste Algorithmus 6) wird in [GS92] beschrieben.

Name	erwartete Laufzeit
1) Berlekamp-Rabin-Algorithmus	$O\left(d^2 \log(d)\, mn^3\right)$
2) Linearisierte-Polynome-Algorithmus	$O\left(m^3n^3 + dm^2n^3 + d^3n^2\right)$
3) Berlekamp-Spur-Algorithmus	$O\left(d^2mn^3 + d^3n^2\right)$
4) modifizierter Lin.-Polynome-Algorithmus	$O\left(dmn^3 + d^3n^2\right)$
5) modifizierter Spur-Algorithmus	$O\left(d^2mn^3\right)$
6) von zur Gathen-Shoup-Algorithmus	$O\left(d\left(\log(d)\right)^{O(1)}\left(dn^2 + mn^3\right)\right)$

Die Algorithmen, die mit linearisierten Polynomen arbeiten, sind insofern von besonderem Interesse, als es sowieso sinnvoll scheint, zunächst ein affines Vielfaches von $U(\varphi) - y$ zu bestimmen, um „gute“ Funktionen φ (d.h. Funktionen, die beispielsweise der Attacke

über affine Vielfache widerstehen, vgl. Abschnitt 5.1.2) zu finden. Ist ein solches affines Vielfaches schon bekannt, fällt bei den oben genannten Algorithmen 2) und 4) der Summand d^3n^2 in den Laufzeiten weg.

Die Laufzeiten der oben genannten Algorithmen haben gemeinsam, dass sie alle quadratisch oder sogar kubisch in d und n sind. Ausgehend von einer Wahl von $d = O(n)$ (wie Patarin sie beispielsweise vorschlägt) ergeben sich für die Algorithmen 1) bis 5) Laufzeiten von ungefähr $O\left(n^5\right)$, d.h. um effizient entschlüsseln und signieren zu können, dürfen n und d nicht zu groß (also beispielsweise ungefähr $d, n \leq 2^8$) gewählt werden.
Insbesondere d kommt immer mindestens als quadratischer Faktor vor, so dass — im Gegensatz zu dem im vorigen Abschnitt beschriebenen Aspekt der Sicherheit — d für die Effizienz eher klein und auf gar keinen Fall $d = 2 \cdot q^{n-1}$ maximal gewählt werden sollte.

2.3.3 Problem der fehlenden Bijektivität von P

Der größte Nachteil von HFE gegenüber C^* ist die fehlende Bijektivität der Einwegfunktion P. Diese kann durch Einfügen von Redundanz und Randomisierung in der Praxis umgangen werden, wie in Abschnitt 2.2 gezeigt wurde. In diesem Abschnitt wird nun untersucht, wieviel Redundanz bzw. Randomisierung nötig ist, um mit hoher Wahrscheinlichkeit eine erfolgreiche Entschlüsselung bzw. Signierung durchführen zu können.

Dabei sind zwei Probleme zu unterscheiden: Beim Signieren ist es ein Problem, wenn P nicht surjektiv ist, es also zu einem $y \in K^n$ kein $x \in K^n$ mit $P(x) = y$ gibt. Beim Entschlüsseln dagegen besteht das Problem darin, dass P im Allgemeinen nicht injektiv ist, zu einem $y \in K^n$ also unter Umständen mehrere Lösungen $x \in K^n$ mit $P(x) = y$ existieren. Im Folgenden wird nun untersucht, mit welchen Wahrscheinlichkeiten diese Problemfälle in der Anwendung jeweils auftreten. Dabei wird davon ausgegangen, dass $y \in K^n$ und P aus allen Funktionen $K^n \to K^n$, die aus HFE-Polynomen vom Grad d wie in Abschnitt 2.1 beschrieben entstehen, zufällig, gemäß der Gleichverteilung gewählt werden.

Um die Wahrscheinlichkeiten einfach beschreiben zu können, wird noch eine Schreibweise benötigt:

Definition 2.3.1
Seien y und P wie oben beschrieben gewählt. Dann sei die Anzahl der Lösungen der Gleichung $P(x) = y$ mit

$$N(P, y) := |\{x \in K^n \mid P(x) = y\}|$$

bezeichnet und für $k \in \{0, \ldots, d\}$ sei

$$N_k := \Pr\left(N(P, y) = k\right).$$

Zunächst werden nun die gesuchten Wahrscheinlichkeiten mit Hilfe der N_k ausgedrückt.

Für das Signieren ist die Situation recht einfach. Die in Abschnitt 2.2.2 beschriebenen HFE-Signaturen nutzen alle das gleiche Schema: Aus der zu signierenden Nachricht wird durch Hinzufügen von Randomisierungen ein $y \in K^n$ gewonnen, zu dem dann ein $x \in K^n$ mit $P(x) = y$ gesucht wird. Gibt es dies nicht, so wird immer wieder ein neues $y \in K^n$ zu der zu signierenden Nachricht bestimmt, solange bis eines gefunden wird, für das eine Lösung x existiert. Von Interesse ist hier also vor allem die Wahrscheinlichkeit, dass zu einem $y \in K^n$ kein x mit $P(x) = y$ existiert. Denn daraus kann sowohl die durchschnittlich benötigte Rundenanzahl als auch die Anzahl an Runden abgeleitet werden, die benötigt wird, um mit einer vorgegebenen Wahrscheinlichkeit ein y zu finden, zu dem ein x existiert. Aus letzterer kann dann bestimmt werden, wie groß die Randomisierung sein muss, die im Schema eingebaut ist, d.h. beispielsweise wie viele Bits von y zufällig gewählt werden müssen.
Um diese Wahrscheinlichkeiten berechnen zu können, wird davon ausgegangen, dass die aus einer Nachricht durch Hinzufügen von Randomisierungen erhaltenen $y \in K^n$ unabhängig und gleichverteilt im K^n sind, wie es beispielsweise bei der „ca.128-Bit-Signatur“ annähernd erreicht wird. Dann gilt folgender Satz:

Satz 2.3.2

i) Die Wahrscheinlichkeit, dass zu einem $y \in K^n$ keine Signatur $x \in K^n$ mit $P(x) = y$ gefunden werden kann, ist gerade N_0.

ii) Durchschnittlich werden $\overline{R} = \frac{1}{1-N_0}$ Runden benötigt, bis zu einer Nachricht eine Signatur gefunden wird.

iii) Um die Fehlerwahrscheinlichkeit auf Ψ zu beschränken, d.h. zu einer zufällig gewählten Nachricht mit Wahrscheinlichkeit $1 - \Psi$ eine Signatur zu finden, müssen $\widehat{R} = \left\lceil \frac{\ln \Psi}{\ln N_0} \right\rceil$ Runden durchgeführt werden.

Beweis:
i) ist klar nach Definition von N_k.
Da nach Annahme zu einer Nachricht in jeder Runde ein neues $y \in K^n$ unabhängig von den vorherigen und zufällig gewählt wird, ist in jeder Runde die Wahrscheinlichkeit, dass ein passendes x gefunden wird, genau $1 - N_0$. Die Zufallsvariable R, die angibt, nach wie vielen Runden das erste x gefunden wird, ist also gemäß der geometrischen Verteilung verteilt, d.h. es ist $\Pr(R = k) = (1 - N_0) N_0^{k-1}$. Der Erwartungswert dieser Verteilung ist gerade $\overline{R} = \frac{1}{1-N_0}$; damit folgt *ii)*. Weiter ist

$$\Pr(R \leq k) = \sum_{i=1}^{k} \Pr(R = i) = 1 - N_0^k.$$

Die Wahrscheinlichkeit, spätestens nach $\widehat{R}$ Runden eine Signatur zu finden, ist also $\Pr\left(R \leq \widehat{R}\right) = 1 - N_0^{\widehat{R}}$. Damit folgt wegen

$$\Pr\left(R \leq \widehat{R}\right) \geq 1 - \Psi \quad \Leftrightarrow \quad N_0^{\widehat{R}} \leq \Psi \quad \Leftrightarrow \quad \widehat{R} \geq \frac{\ln \Psi}{\ln N_0}$$

Behauptung *iii*). ■

Um entsprechende Aussagen für das Entschlüsseln treffen zu können, muss zunächst die eingefügte Redundanz formalisiert werden. Dazu wird definiert:

Definition 2.3.3
Sei $x \in K^n$ zufällig und gleichverteilt gewählt. Dann sei mit ρ die Wahrscheinlichkeit bezeichnet, dass x die Redundanzkriterien erfüllt (die unter Umständen noch von dem wie oben beschrieben gewählten y abhängen).

Wird innere Redundanz verwendet, sind die Redundanzkriterien also unabhängig von y, so ist $\rho = \frac{|\{x \in K^n \mid x \text{ erfüllt Redundanz}\}|}{|K^n|}$. Falls beispielsweise ein fehlererkennender Code verwendet wird, entspricht ρ also gerade dessen Informationsrate. Wird äußere Redundanz verwendet, wird also zusätzlich zu $y = P(x)$ noch ein Kontrollwert aus einem Raum $\mathcal{K}$ (etwa ein Hashwert) verschickt, kann ρ ausgedrückt werden als $\rho = \frac{1}{|\mathcal{K}|}$.
Zudem sei analog zur obigen Definition von $N(P, y)$ noch

$$\widetilde{N}(P, y) := |\{x \in K^n \mid P(x) = y,\ x \text{ erfüllt die Redundanz}\}|$$

definiert. Um diese Werte in Abhängigkeit von $N(P, y)$ abschätzen zu können, sei ferner angenommen, dass die Lösungen von $P(x) = y$ unabhängig voneinander, in K^n gleichverteilt sind.

Der Problemfall beim Entschlüsseln ist — wie oben beschrieben — eine nicht eindeutige Entschlüsselung. Es wird also nach der Wahrscheinlichkeit gesucht, dass $\widetilde{N}(P, y) > 1$ ist. Allerdings ist beim Entschlüsseln y immer von der Art, dass es mindestens eine Lösung x für $P(x) = y$ gibt, die die Redundanzkriterien erfüllt. Es gilt also immer $\widetilde{N}(P, y) \geq 1$. Die gesuchte Fehlerwahrscheinlichkeit ist also die bedingte Wahrscheinlichkeit $\Pr\left(\widetilde{N}(P, y) > 1 \mid \widetilde{N}(P, y) \geq 1\right)$. Dafür gilt

Satz 2.3.4
Die Wahrscheinlichkeit, dass ein Chiffretext $y \in K^n$, der zu einer Nachricht x gehört, nicht eindeutig entschlüsselt werden kann, ist

$$\Pr\left(\widetilde{N}(P, y) > 1 \mid \widetilde{N}(P, y) \geq 1\right) \ \leq \ \omega_d \frac{\rho}{(1-\rho)^{d-1}} \qquad \textit{mit } \omega_d = \frac{\sum_{k=2}^{d} \left(2^k - k - 1\right) N_k}{\sum_{k=1}^{d} k \cdot N_k}.$$

Um mit Wahrscheinlichkeit $1 - \Psi$ *eindeutig entschlüsseln zu können, genügt es, soviel Redundanz zu verwenden, dass*

$$\rho \leq \frac{\Psi}{\omega_d + (d-1)\,\Psi}$$

gilt.

Beweis:

Unter der obigen Annahme über die Verteilung der $k = N(P,y)$ Nullstellen von $P(x) = y$ ist die Wahrscheinlichkeit, dass genau $\widetilde{k}$ davon die Redundanz erfüllen

$$\Pr\left(\widetilde{N}(P,y) = \widetilde{k} \mid N(P,y) = k\right) = \binom{k}{\widetilde{k}} \rho^{\widetilde{k}} (1-\rho)^{k-\widetilde{k}}.$$

Also ist

$$\begin{aligned}\Pr\left(\widetilde{N}(P,y) = \widetilde{k}\right) &= \sum_{k=0}^{d} \underbrace{\Pr\left(N(P,y) = k\right)}_{N_k} \cdot \underbrace{\Pr\left(\widetilde{N}(P,y) = \widetilde{k} \mid N(P,y) = k\right)}_{=0,\text{ für } k<\widetilde{k}} \\ &= \sum_{k=\widetilde{k}}^{d} N_k \binom{k}{\widetilde{k}} \rho^{\widetilde{k}} (1-\rho)^{k-\widetilde{k}}.\end{aligned}$$

Damit kann die gesuchte Wahrscheinlichkeit ausgedrückt werden als

$$\begin{aligned}\Pr\left(\widetilde{N}(P,y) > 1 \mid \widetilde{N}(P,y) \geq 1\right) &= \frac{\Pr\left(\widetilde{N}(P,y) > 1\right)}{\Pr\left(\widetilde{N}(P,y) \geq 1\right)} = \frac{\sum_{\widetilde{k}=2}^{d} \Pr\left(\widetilde{N}(P,y) = \widetilde{k}\right)}{\sum_{\widetilde{k}=1}^{d} \Pr\left(\widetilde{N}(P,y) = \widetilde{k}\right)} \\ = \left(1 + \frac{\Pr\left(\widetilde{N}(P,y) = 1\right)}{\sum_{\widetilde{k}=2}^{d} \Pr\left(\widetilde{N}(P,y) = \widetilde{k}\right)}\right)^{-1} &= \left(1 + \frac{\sum_{k=1}^{d} N_k \cdot k \cdot \rho(1-\rho)^{k-1}}{\sum_{\widetilde{k}=2}^{d} \sum_{k=\widetilde{k}}^{d} N_k \binom{k}{\widetilde{k}} \rho^{\widetilde{k}} (1-\rho)^{k-\widetilde{k}}}\right)^{-1} \\ &= \left(1 + \frac{\rho \sum_{k=1}^{d} N_k \cdot k \cdot (1-\rho)^{k-1}}{\sum_{k=2}^{d} N_k \sum_{\widetilde{k}=2}^{k} \binom{k}{\widetilde{k}} \rho^{\widetilde{k}} (1-\rho)^{k-\widetilde{k}}}\right)^{-1}.\end{aligned}$$

Dieser Ausdruck kann wegen $0 \leq \rho \leq 1$ nach oben abgeschätzt werden durch

$$\begin{aligned}\left(1 + \frac{\rho(1-\rho)^{d-1} \cdot \sum_{k=1}^{d} k \cdot N_k}{\rho^2 \cdot \sum_{k=2}^{d} N_k \sum_{\widetilde{k}=2}^{k} \binom{k}{\widetilde{k}}}\right)^{-1} &= \left(1 + \frac{(1-\rho)^{d-1}}{\rho} \cdot \frac{\sum_{k=1}^{d} k \cdot N_k}{\sum_{k=2}^{d} (2^k - k - 1) N_k}\right)^{-1} \\ &= \left(1 + \frac{(1-\rho)^{d-1}}{\omega_d \rho}\right)^{-1} = \frac{\omega_d \rho}{\omega_d \rho + (1-\rho)^{d-1}} \leq \frac{\omega_d \rho}{(1-\rho)^{d-1}}.\end{aligned}$$

Falls nun $\rho \leq \frac{\Psi}{\omega_d + (d-1)\Psi}$ gilt, ist weiter

$$\begin{aligned}\Pr\left(\widetilde{N}(P,y) > 1 \mid \widetilde{N}(P,y) \geq 1\right) &\leq \frac{\omega_d \rho}{(1-\rho)^{d-1}} \\ &\leq \frac{\omega_d \rho}{1 - (d-1)\rho} = \frac{\omega_d}{\frac{1}{\rho} - (d-1)} \leq \frac{\omega_d}{\frac{\omega_d + (d-1)\Psi}{\Psi} - (d-1)} = \Psi.\end{aligned}$$

■

Die beiden Sätze 2.3.2 und 2.3.4 zeigen, dass in die gesuchten Fehlerwahrscheinlichkeiten im Wesentlichen nur die Wahrscheinlichkeiten N_k eingehen. Leider sind für diese Wahrscheinlichkeiten keine exakten Werte bekannt. Deshalb wird im Folgenden untersucht, was eine sinnvolle Annahme für diese Werte sein könnte.

Patarin bemerkt in [Pat96a], dass es „meist nur 0,1 oder 2 Lösungen“ gäbe (dass also N_0, N_1 und N_2 deutlich größer als alle anderen N_k wären), ohne allerdings irgendwelche Hinweise dafür zu geben, warum dies so sein könnte. In [CGP01] stellt er dann sogar die Behauptung auf, dass es möglich sei, $N_0 \approx \frac{1}{e}$ zu beweisen. Später korrigierte er diese Behauptung aber dahingehend, dass es nur eine Vermutung aufgrund von Simulationen und Überlegungen sei, ähnlich wie sie hier im Folgenden auch beschrieben werden.

In den Testreihen, die in Abschnitt 3.4.2 näher erläutert werden, wurden insgesamt 75599 zufällige HFE-Systeme für verschiedene Parameter n, q und d erzeugt und unter anderem die Anzahl der Lösungen dieser Systeme bestimmt. Die nun folgende Tabelle gibt eine Übersicht darüber, wie viele dieser Systeme eine bestimmte Anzahl an Lösungen hatten:

k	0	1	2	3	4	≥ 5
Systeme mit genau k Lösungen	27710	28012	13852	4565	1210	250
Anteil	0,3665	0,3705	0,1832	0,0604	0,0160	0,0033

Verglichen mit der Tabelle

k	0	1	2	3	4
$\frac{1}{k!e}$	0,3679	0,3679	0,1839	0,0613	0,0153

führen diese Beobachtungen zu folgender Vermutung:

Vermutung 2.3.5
Für die Wahrscheinlichkeit N_k, dass ein zufällig gewähltes HFE-System $P(x) = y$ genau k Lösungen hat, gilt

$$N_k \approx \frac{1}{k!e}.$$

Die Zufallsvariable $N(P, y)$ ist also annähernd poissonverteilt mit Parameter $\lambda = 1$.

Da — wie oben schon erwähnt — leider kein Beweis dieser Vermutung bekannt ist, wird im Folgenden nun beschrieben, was neben den Ergebnissen der oben beschriebenen Simulationen noch dafür spricht, dass diese Vermutung korrekt ist.

Dazu wird zunächst gezeigt, dass anstelle der Anzahl $N(P, y)$ der Lösungen von $P(x) = y$ mit zufällig gewählten P und y genauso gut die Anzahl der Nullstellen der in P versteckten univariaten Abbildung φ betrachtet werden kann. Dazu bezeichne $N(u)$ die Anzahl der verschiedenen Nullstellen des univariaten Polynoms $u \in L[x]$ in L. Mit dieser Bezeichnung gilt

Satz 2.3.6

Sei $M \subseteq \mathcal{M}_{n,\delta}$ eine Teilmenge der multivariaten Polynomsysteme mit n Variablen und n Gleichungen und sei

$$U := \left\{ U(\varphi) \mid \varphi : L \to L,\ \varphi = \phi^{-1} \circ t^{-1} \circ P \circ s^{-1} \circ \phi \text{ für ein } P \in M \right\} \subseteq \mathcal{U}_{n,\delta}$$

die Menge der (bezüglich affiner Transformationen s und t und eines K-Isomorphismus ϕ) zu M gehörigen, univariaten Polynome. Dann gilt für alle $y \in K^n$ und alle $k \in \mathbb{N}_0$

$$\left|\{P : K^n \to K^n \mid M(P) \in M,\ N(P,y) = k\}\right| = \left|\left\{u \in U \mid N\left(u - \phi^{-1}\left(t^{-1}(y)\right)\right) = k\right\}\right|.$$

Ist für $a \in L$ und $u \in U$ auch $u + a \in U$, so ist diese Anzahl gerade gleich

$$\left|\{u \in U \mid N(u) = k\}\right|,$$

also insbesondere unabhängig von y.

Beweis:

Da s, t und ϕ bijektiv sind, existiert zu jedem Polynom $u = U(\varphi) \in U$ genau ein $P = t \circ \phi \circ \varphi \circ \phi^{-1} \circ s$ mit $M(P) \in M$ und für alle $x, y \in K^n$ gilt

$$P(x) = y \ \Leftrightarrow\ \varphi\left(\phi^{-1}(s(x))\right) = \phi^{-1}\left(t^{-1}(y)\right) \ \Leftrightarrow\ u\left(\phi^{-1}(s(x))\right) - \phi^{-1}\left(t^{-1}(y)\right) = 0.$$

Da zudem $\phi^{-1} \circ s : K^n \to L$ bijektiv ist, folgt die erste Gleichung. Ist nun für $a \in L$ und $u \in U$ auch $u + a \in U$, so bildet die injektive Abbildung

$$U \to L[x],\ u \mapsto \underbrace{u - \phi^{-1}\left(t^{-1}(y)\right)}_{\in L}$$

gerade U auf sich selber ab und daraus folgt die Behauptung. ■

Die Verteilung von $N(P,y)$ für alle HFE-Systeme P und $y \in K^n$ entspricht also genau der Verteilung der Anzahlen der Nullstellen $N(\varphi)$ der HFE-Polynome φ vom Grad $\leq d$, also der Polynome aus

$$\mathcal{H}_d := \left\{ \varphi(x) = \sum_{i,j} \beta_{ij} x^{q^i + q^j} + \sum_i \alpha_i x^{q^i} + \mu_0 \,\middle|\, \beta_{ij}, \alpha_i, \mu_0 \in L,\ grad(\varphi) \leq d \right\} \subseteq \mathcal{U}_{n,2},$$

da diese Menge aufgrund des frei wählbaren absoluten Gliedes $\mu_0 \in L$ die im Satz geforderte Eigenschaft hat.

Auch in dieser Form ist die Verteilung nicht exakt bekannt, aber im Folgenden werden zwei Beobachtungen beschrieben, die darauf hinweisen, dass diese Verteilung wirklich annähernd eine Poisson-Verteilung ist.

Der erste Hinweis ergibt sich aus der Betrachtung einer Obermenge von $\mathcal{H}_d$, nämlich der Menge $\mathcal{P}_d := \{f \in L[x] \mid grad(f) \leq d\}$ aller univariaten Polynome vom Grad $\leq d$. Für diese Menge lässt sich die entsprechende Verteilung nämlich exakt berechnen.
Dafür werden zunächst noch einige Bezeichnungen benötigt:

Definition 2.3.7

- *Zu $f \in L[x]$ sei im Folgenden $f = c \cdot \prod f_i$, $c \in L$, $f_i \in L[x]$ die Zerlegung in nicht notwendig verschiedene, irreduzible Faktoren.*
- *Für $\kappa, d, \delta, l \in \mathbb{N}_0$ seien*

$$L_\kappa(d) := |\{f \in L[x] \mid grad(f) \leq d,\ f \text{ normiert},\ |\{f_i \mid grad(f_i) = 1\}| = \kappa\}|$$

und

$$L_\kappa(\delta, l) := \left| \left\{ f \in L[x] \;\middle|\; \begin{array}{c} grad(f) = \delta,\ |\{f_i \mid grad(f_i) = 1\}| = \kappa, \\ f \text{ normiert},\ \max(grad(f_i)) = l \end{array} \right\} \right|$$

Aus dieser Definition folgt offensichtlich folgende Beziehung:

Lemma 2.3.8
Es gilt

$$L_\kappa(d) = \sum_{\delta=0}^{d} \sum_{l=0}^{\delta} L_\kappa(\delta, l).$$

Beweis:
Die für $L_\kappa(d)$ abgezählte Menge kann disjunkt zerlegt werden bezüglich des Grades $\delta = grad(f) \in \{0, \ldots, d\}$ und des maximalen in der Zerlegung von f auftretenden Grades $l = \max(grad(f_i))$. Dies ergibt die Behauptung. ■

Um die Werte $L_\kappa(\delta, l)$ berechnen zu können, wird noch die Anzahl $Irr(\delta)$ der irreduziblen, normierten Polynome des Grades δ in $L[x]$ benötigt. Für diese Anzahl gilt nach [GG99] folgendes Lemma:

Lemma 2.3.9
Mit der Möbiusfunktion

$$\mu(\delta) = \begin{cases} 1, & \text{falls } \delta = 1 \text{ gilt,} \\ (-1)^i, & \text{falls } \delta \text{ Produkt genau } i \text{ verschiedener Primzahlen ist,} \\ 0, & \text{falls } \delta \text{ nicht quadratfrei ist} \end{cases}$$

gilt für die Anzahl $Irr(\delta)$ der irreduziblen, normierten Polynome vom Grad δ aus $L[x]$

$$Irr(\delta) = \frac{1}{\delta} \sum_{\delta' | \delta} \mu\left(\frac{\delta}{\delta'}\right) |L|^{\delta'}.$$

Damit kann folgende Rekursionsformel für die $L_\kappa(\delta, l)$ formuliert werden:

Satz 2.3.10

Seien $\kappa, \delta, l \in \mathbb{N}_0$.

i) Für $\delta > \kappa$ *und* $l \geq 2$ *ist*

$$L_\kappa(\delta, l) = \sum_{j=1}^{\lfloor \frac{\delta}{l} \rfloor} \binom{Irr(l) + j - 1}{j} \cdot \sum_{l'=0}^{l-1} L_\kappa(\delta - jl, l') .$$

ii) Abhängig von κ *können die Startwerte für die Rekursion berechnet werden durch*

$$L_0(\delta, l) = \begin{cases} 1 & \text{für } \delta = l = 0, \\ 0 & \text{für } \delta = 0,\ l > 0, \\ 0 & \text{für } \delta > 0,\ l \in \{0, 1\}, \end{cases}$$

bzw. für $\kappa \geq 1$ *durch*

$$L_\kappa(\delta, l) = \begin{cases} 0 & \text{für } \delta < \kappa, \\ \binom{|L|^\delta + \kappa - 1}{\kappa} & \text{für } \delta = \kappa,\ l = 1, \\ 0 & \text{für } \delta = \kappa,\ l \neq 1, \\ 0 & \text{für } \delta > \kappa,\ l \in \{0, 1\}. \end{cases}$$

Beweis:

zu *i*): Sei f ein Polynom aus der Menge, die durch $L_\kappa(\delta, l)$ abgezählt wird, und dazu $j = |\{f_i \mid grad(f_i) = l\}|$. Für die j irreduziblen Faktoren f_i von f, die genau den Grad l haben, gibt es also gerade $\binom{Irr(l)+j-1}{j}$ Möglichkeiten, diese aus den $Irr(l)$ möglichen irreduziblen Polynomen vom Grad l zu wählen. Die restlichen Faktoren ergeben dann gerade ein Polynom $\frac{f}{\prod_{grad(f_i)=l} f_i}$ vom Grad $\delta - jl$, dessen irreduzible Faktoren alle einen Grad $< l$, d.h. aus $\{0, \ldots, l-1\}$, haben. Zu jeder der $\binom{Irr(l)+j-1}{j}$ möglichen Wahlen für die Faktoren vom Grad l gibt es also nochmal $\sum_{l'=0}^{l-1} L_\kappa(\delta - jl, l')$ mögliche Wahlen für die restlichen Faktoren. Aufsummiert über alle möglichen Anzahlen j der Faktoren von maximalem Grad folgt die Behauptung.

ii) folgt direkt aus Definition 2.3.7 durch Betrachtung der verschiedenen Fälle. ■

Mit Hilfe von Satz 2.3.10 kann also der Anteil $\frac{L_\kappa(d) \cdot |L|}{|\mathcal{P}_d|} = \frac{L_\kappa(d)}{|L|^d}$ der Polynome aus $\mathcal{P}_d \subset L[x]$ bestimmt werden, die genau κ Linearfaktoren besitzen. Beispielsweise für $L = \mathbb{F}_{2^6}$ ergeben sich die in der folgenden Tabelle zusammengefassten Anteile:

d \ κ	0	1	2	3	4	5	6
2	0,4922	0	0,5078	0	0	0	0
3	0,3333	0,4922	0	0,1746	0	0	0
4	0,3711	0,3333	0,2499	0	0,0457	0	0
5	0,3640	0,3711	0,1692	0,0859	0	0,0097	0
6	0,3651	0,3640	0,1885	0,0582	0,0225	0	0,0017
7	0,3650	0,3651	0,1849	0,0648	0,0152	0,0048	0
8	0,3650	0,3650	0,1854	0,0635	0,0170	0,0032	0,0009
9	0,3650	0,3650	0,1853	0,0637	0,0166	0,0036	0,0006

Für $L = \mathbb{F}_{2^{15}}$ ergibt sich folgende Tabelle:

d \ κ	0	1	2	3	4	5	6
2	0,5000	0	0,5000	0	0	0	0
3	0,3333	0,5000	0	0,1667	0	0	0
4	0,3750	0,3333	0,2500	0	0,0417	0	0
5	0,3667	0,3750	0,1667	0,0833	0	0,0083	0
6	0,3681	0,3667	0,1875	0,0556	0,0208	0	0,0014
7	0,3679	0,3681	0,1833	0,0625	0,0139	0,0042	0
8	0,3679	0,3679	0,1840	0,0611	0,0156	0,0028	0,0007
9	0,3679	0,3679	0,1839	0,0613	0,0153	0,0031	0,0005

Aus diesen Werten und auch aus weiteren Berechnungen bezüglich anderer Körper L lässt sich leicht erkennen, dass (falls L und d nicht besonders klein sind) der Anteil der Polynome in $\mathcal{P}_d$, die genau κ Linearfaktoren haben, ungefähr $\frac{1}{\kappa! e}$ beträgt.

Gesucht war ursprünglich allerdings die Verteilung der Nullstellenanzahlen der Polynome aus $\mathcal{P}_d$ (ohne Vielfachheiten gezählt, d.h. die Verteilung der Anzahlen der verschiedenen Linearfaktoren). Dazu folgendes Lemma:

Lemma 2.3.11
Sei für $k > 0$

$$A_{k,\kappa} := \frac{|\{f \in \mathcal{P}_d \mid N(f) = k\}|}{|\{f \in \mathcal{P}_d \mid f \text{ hat genau } \kappa \text{ Linearfaktoren}\}|}$$

der Anteil der Polynome in $\mathcal{P}_d$, die genau k verschiedene Nullstellen haben, an den Polynomen, die insgesamt κ Linearfaktoren besitzen. Dann gilt

$$A_{k,\kappa} = \frac{\binom{|L|}{k}\binom{\kappa-1}{\kappa-k}}{\binom{|L|+\kappa-1}{\kappa}}.$$

Beweis:
Es gibt $|L|$ verschiedene Linearfaktoren und somit insgesamt $\binom{|L|+\kappa-1}{\kappa}$ Möglichkeiten, daraus κ nicht notwendig verschiedene Linearfaktoren auszuwählen.
Andererseits gibt es $\binom{|L|}{k}$ Möglichkeiten, k verschiedene Linearfaktoren auszuwählen. Für jede solche Auswahl gibt es dann $\binom{k+(\kappa-k)-1}{\kappa-k} = \binom{\kappa-1}{\kappa-k}$ Möglichkeiten, die restlichen $\kappa - k$ Linearfaktoren aus diesen auszuwählen. Daraus folgt die Behauptung. ■

Damit kann der gesuchte Anteil der Polynome mit $k \geq 1$ verschiedenen Nullstellen in $\mathcal{P}_d$ berechnet werden als

$$\sum_{\kappa=k}^{d} A_{k,\kappa} \frac{L_\kappa(d)}{|L|^d}.$$

Asymptotisch, d.h. für große $|L|$ betrachtet, ist gerade

$$A_{k,\kappa} = |L|^{k-\kappa} \cdot \frac{\kappa!}{k!}\binom{\kappa-1}{\kappa-k} \approx \begin{cases} 0 & \text{für } k < \kappa, \\ 1 & \text{für } k = \kappa, \end{cases} \quad \text{d.h.} \quad \sum_{\kappa=k}^{d} A_{k,\kappa} \frac{L_\kappa(d)}{|L|^d} \approx \frac{L_k(d)}{|L|^d}.$$

Der Unterschied zwischen den bisher berechneten Werten und den gesuchten Anteilen kann für große $|L|$ also vernachlässigt werden. Dies wird auch dadurch verdeutlicht, dass sich für $L = \mathbb{F}_{2^{15}}$ exakt die oben angegebene Tabelle auch für die durch die $A_{k,\kappa}$ korrigierten Werte ergibt. Schon für $L = \mathbb{F}_{2^6}$ ergeben sich die folgenden Werte, die sich kaum von den obigen unterscheiden:

d \ κ	0	1	2	3	4	5	6
2	0,4922	0,0156	0,4922	0	0	0	0
3	0,3333	0,4924	0,0154	0,1589	0	0	0
4	0,3711	0,3410	0,2426	0,0075	0,0379	0	0
5	0,3640	0,3765	0,1716	0,0785	0,0023	0,0071	0
6	0,3651	0,3699	0,1880	0,0566	0,0187	0,0006	0,0011
7	0,3650	0,3709	0,1850	0,0616	0,0138	0,0035	0,0001
8	0,3650	0,3708	0,1854	0,0607	0,0149	0,0026	0,0005
9	0,3650	0,3708	0,1854	0,0608	0,0147	0,0028	0,0004

Die hier berechneten Werte liefern also einen starken Hinweis dafür, dass Vermutung 2.3.5 zutrifft. Diese Anteile können aber auf diese Weise nur für $\mathcal{P}_d$ und nicht für $\mathcal{H}_d$ berechnet werden und der Zusammenhang zwischen den Werten für $\mathcal{P}_d$ und $\mathcal{H}_d$ ist leider nicht bekannt.

Ein weiterer Hinweis kann durch eine Rechnung, die sich direkt auf die Menge $\mathcal{H}_d$ bezieht, erhalten werden. Für diese Menge können nämlich zumindest der Erwartungswert und die Varianz der zugrunde liegenden Verteilung auf eine andere Art und Weise exakt berechnet werden:

Satz 2.3.12
Die Zufallsvariable $N : \mathcal{H}_d \to \mathbb{N}_0,\ \varphi \mapsto |\{a \in L \mid \varphi(a) = 0\}|$ *hat unter Annahme der Gleichverteilung auf* $\mathcal{H}_d$ *den Erwartungswert*

$$\mathbb{E}(N) = \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi) = 1$$

bei einer Varianz von

$$\mathbb{V}(N) = \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} (N(\varphi) - 1)^2 = 1 - \frac{1}{|L|}.$$

Beweis:
Es ist $N(\varphi) = \sum_{\substack{a \in L \\ \varphi(a)=0}} 1$, also lässt sich der Erwartungswert auch schreiben als

$$\mathbb{E}(N) = \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi) = \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} \sum_{\substack{a \in L \\ \varphi(a)=0}} 1 = \frac{1}{|\mathcal{H}_d|} \sum_{a \in L} \sum_{\substack{\varphi \in \mathcal{H}_d \\ \varphi(a)=0}} 1.$$

Betrachte nun zu jedem $a \in L$ den Einsetzunghomomorphismus $\pi_a : \mathcal{H}_d \to L,\ \varphi \mapsto \varphi(a)$. Dann ist

$$|\{\varphi \in \mathcal{H}_d \mid \varphi(a) = 0\}| = |Kern(\pi_a)| = \frac{|\mathcal{H}_d|}{|L|},$$

der Erwartungswert lässt sich mit der obigen Formel also schreiben als

$$\mathbb{E}(N) = \frac{1}{|\mathcal{H}_d|} \sum_{a \in L} \sum_{\substack{\varphi \in \mathcal{H}_d \\ \varphi(a)=0}} 1 = \frac{1}{|\mathcal{H}_d|} \sum_{a \in L} \frac{|\mathcal{H}_d|}{|L|} = 1.$$

Für die Varianz betrachte zunächst ähnlich wie oben

$$\frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi)^2 = \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} \sum_{\substack{a \in L \\ \varphi(a)=0}} \sum_{\substack{b \in L \\ \varphi(b)=0}} 1 = \frac{1}{|\mathcal{H}_d|} \sum_{a,b \in L} \sum_{\substack{\varphi \in \mathcal{H}_d \\ \varphi(a)=\varphi(b)=0}} 1.$$

Die Bedingungen $\varphi(a) = 0$ und $\varphi(b) = 0$ beschreiben zwei lineare Gleichungen für die Koeffizienten von φ. Diese sind für $a \neq b$ linear unabhängig, die Anzahl „passender“ φ wird durch sie also auf $\frac{|\mathcal{H}_d|}{|L|^2}$ beschränkt, und für $a = b$ sind sie natürlich gleich, so dass in diesem Fall analog zur Berechnung des Erwartungswertes $\frac{|\mathcal{H}_d|}{|L|}$ passende Polynome existieren. Es ist also

$$\frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi)^2 = \frac{1}{|\mathcal{H}_d|} \left(\sum_{\substack{a \in L \\ b \in L \setminus \{a\}}} \frac{|\mathcal{H}_d|}{|L|^2} + \sum_{a \in L} \frac{|\mathcal{H}_d|}{|L|} \right) = \frac{|L| \cdot (|L| - 1)}{|L|^2} + 1 = 2 - \frac{1}{|L|}.$$

Damit gilt für die Varianz

$$\begin{aligned} \mathbb{V}(N) &= \frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} (N(\varphi) - 1)^2 \\ &= \underbrace{\frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi)^2}_{=2-\frac{1}{|L|}} - 2 \cdot \underbrace{\frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} N(\varphi)}_{=\mathbb{E}(N)=1} + \underbrace{\frac{1}{|\mathcal{H}_d|} \sum_{\varphi \in \mathcal{H}_d} 1^2}_{=1} = 1 - \frac{1}{|L|}. \end{aligned}$$

■

Bemerkung 2.3.13
Der Erwartungswert lässt sich auch schreiben als $\mathbb{E}(N) = \sum \Pr(N(\varphi) = k) \cdot k = \sum k \cdot N_k$. *Damit vereinfacht dieser Satz das* ω_d *in Satz 2.3.4 zu*

$$\omega_d = \frac{\sum_{k=2}^{d} \left(2^k - k - 1\right) N_k}{\mathbb{E}(N)} = \sum_{k=2}^{d} \left(2^k - k - 1\right) \cdot N_k.$$

Die in Satz 2.3.12 berechneten Parameter der Verteilung von N liefern einen weiteren Hinweis darauf, dass die $N(P, y)$ annähernd poissonverteilt sind. Denn eine Poisson-Verteilung zum Parameter $\lambda = 1$ hat gerade den Erwartungswert und die Varianz 1.
Zudem zeigt der Satz aber auch, dass den $N(P, y)$ keine *exakte* Poisson-Verteilung zugrunde liegen kann, da die berechnete Varianz nicht exakt 1 ist.

Weiter kann mit Hilfe dieses Satzes und der Tschebyscheff-Ungleichung folgende Abschätzung aufgestellt werden:

$$\Pr(N(P, y) \leq 2) \quad \geq \quad \frac{3}{4}.$$

Dies liefert also in gewisser Weise eine Bestätigung für die Aussage aus [Pat96a], dass es meist $0, 1$ oder 2 Lösungen gäbe, allerdings mit der fragwürdigen Deutung von „meist“ als „in 3 von 4 Fällen“.

Unter der Annahme, dass die Vermutung 2.3.5 wahr ist, genauer gesagt unter der Annahme $N_k = \frac{1}{k!e}$ für alle k, ergibt sich insgesamt das folgende Theorem:

Theorem 2.3.14
Mit $N_k = \frac{1}{k!e}$ für $k = 0, \ldots, d$ gilt das Folgende:

i) Durchschnittlich werden $\overline{R} = \frac{e}{e-1} < 2$ Runden benötigt, bis zu einer Nachricht eine Signatur gefunden wird.

ii) Um mit Wahrscheinlichkeit $1-\Psi$ eine Signatur zu einer Nachricht zu finden, müssen $\widehat{R} = \lceil \ln \frac{1}{\Psi} \rceil$ Runden durchgeführt werden.

iii) Die Fehlerwahrscheinlichkeit beim Entschlüsseln ist

$$\Pr\left(\widetilde{N}(P, y) > 1 \mid \widetilde{N}(P, y) \geq 1\right) \leq \frac{(e-2)\,\rho}{(1-\rho)^{d-1}}.$$

iv) Für $\Psi \leq \frac{3-e}{d-1}$ genügt es, $\rho = \Psi$ zu wählen, um mit Wahrscheinlichkeit $1-\Psi$ eindeutig entschlüsseln zu können.

Beweis:
i) und *ii*) folgen direkt aus Satz 2.3.2 durch Einsetzen der Annahme.
zu *iii*) und *iv*): Nach Bemerkung 2.3.13 gilt unter der Annahme $N_k = \frac{1}{k!e}$

$$\begin{aligned}\omega_d &= \sum_{k=2}^{d} \frac{1}{k!e}\left(2^k - k - 1\right) = \sum_{k=0}^{d} \frac{1}{k!e}\left(2^k - k - 1\right) \\ &= \frac{1}{e}\left(\sum_{k=0}^{d} \frac{2^k}{k!} - \sum_{k=1}^{d} \frac{1}{(k-1)!} - \sum_{k=0}^{d} \frac{1}{k!}\right), \quad \text{also } \lim_{d\to\infty} \omega_d = \frac{e^2 - e - e}{e} = e - 2.\end{aligned}$$

Da zudem $\omega_d - \omega_{d-1} = \frac{1}{d!e}\left(2^d - d - 1\right) \geq 0$ ist, folgt $\omega_d \leq e - 2$ und zusammen mit Satz 2.3.4 somit Behauptung *iii*). Zudem genügt es damit, ebenfalls nach diesem Satz, $\rho \leq \frac{\Psi}{e-2+(d-1)\Psi}$ zu wählen, um die Fehlerwahrscheinlichkeit auf Ψ zu beschränken. Für $\Psi \leq \frac{3-e}{d-1}$ ist aber gerade $\frac{\Psi}{e-2+(d-1)\Psi} \geq \frac{\Psi}{e-2+3-e} = \Psi$. Es genügt also, $\rho = \Psi$ zu wählen und damit folgt Behauptung *iv*). ■

Für eine zu vernachlässigende Fehlerwahrscheinlichkeit von beispielsweise $\Psi = 2^{-80}$ liefert dieses Theorem folgende Werte, die für Redundanz bzw. Randomisierung benötigt werden: Beim Signieren muss das Schema so aufgebaut werden, dass zumindest $\widehat{R} = \lceil 80 \cdot \ln 2 \rceil = 56$ Runden durchgeführt werden können. Es würde also beispielsweise genügen, $\lceil \log_2 56 \rceil = 6$ zusätzliche Bits zu verwenden, die zufällig gewählt werden dürfen. Eine ähnliche Wahl (nämlich 7 zusätzliche Bits) wird auch bei dem von Patarin vorgeschlagenen, auf HFE aufbauenden Kryptosystem QUARTZ verwendet (siehe [CGP01]).
Beim Entschlüsseln dagegen ist die Situation deutlich schlechter. Es muss $\rho = \Psi$ $(= 2^{-80})$ erreicht werden. Wenn äußere Redundanz verwendet werden soll, muss also zu jeder verschlüsselten Nachricht ein zusätzlicher Kontrollwert von 80 Bit Länge mitgeschickt werden. Auch im Falle innerer Redundanz ergibt sich keine bessere Situation. Um beispielsweise Nachrichtenblöcke von 80 Bit so zu verschlüsseln, dass sie mit Wahrscheinlichkeit $\geq 1 - 2^{-80}$ eindeutig entschlüsselt werden können, müsste $n = 80 + 80$ gewählt werden. Diese Verschlüsselung hätte also sozusagen eine „Informationsrate“ von nur $\frac{1}{2}$.

Die Berechnungen in diesem Abschnitt zeigen, dass das HFE-Schema in der Praxis wohl eher als Signatur-Schema geeignet ist. Diese Beobachtung wird auch dadurch bestätigt, dass bislang an konkreten, auf HFE aufbauenden, praktischen Schemata nur das Signatur-Schema QUARTZ vorgeschlagen wurde.

2.3.4 Überblick

In diesem Abschnitt wird nun noch kurz zusammengefasst, was bei der Wahl der Parameter eines HFE-Systems zu beachten ist:

- Natürlich sollten q und n so gewählt werden, dass $|L| = q^n > 2^{80}$ ist, um eine vollständige Suche auf L bzw. K^n zu verhindern.
- Der Grad d des zugrunde liegenden HFE-Polynoms darf nicht zu groß gewählt werden, um φ noch effizient umkehren zu können (vgl. Abschnitt 2.3.2). Andererseits wird in Abschnitt 2.3.1 begründet, dass d nicht zu klein gewählt werden sollte, damit die öffentliche Funktion P einer zufälligen Funktion möglichst ähnlich ist. Dieser Aspekt wird auch in den Abschnitten 3.4.2 und 5.1.3 noch einmal aufgegriffen. Diese

Ergebnisse zusammen ergeben, dass für d ein Wert zwischen 129 und 512 vermutlich sowohl sicher als auch effizient ist.

- Zudem wird in Kapitel 5 gezeigt, dass bei der Wahl von φ beachtet werden muss, dass gewisse Attacken verhindert werden, wie etwa die Attacke über affine Vielfache. Dafür sollte also zumindest vor der Verwendung eines bestimmten φ überprüft werden, ob dieses φ ein kleines affines Vielfaches besitzt (vgl. Abschnitt 5.1.2 und [Fel01]).

- Beim Entschlüsseln scheint eine Redundanz von ungefähr 80 Bit ausreichend zu sein, um sicher eindeutig entschlüsseln zu können, während beim Signieren schon ca. 6 zusätzliche Randomisierungsbits ausreichen, um sicher eine Signatur finden zu können (vgl. Abschnitt 2.3.3).

2.4 Perturbationen

In diesem Abschnitt werden vier Operationen vorgestellt, mit denen das HFE-Kryptosystem und auch andere ähnlich aufgebaute Systeme wie beispielsweise C^* leicht abgeändert werden können, um ihre Sicherheit (hoffentlich) zu erhöhen.
Die Idee all dieser Variationen ist, die klare algebraische Struktur, die dem Verfahren zugrunde liegt, durch ein paar kleine Eingriffe zu zerstören. Dadurch werden Angriffe, die die algebraische Struktur ausnutzen, erschwert, ohne dabei das eigentliche Verfahren stark zu beeinträchtigen. Deshalb werden diese Variationen auch als Perturbationen, d.h. Störungen, bezeichnet.

2.4.1 Verbergen von öffentlichen Polynomen („ − “)

Die erste Variation, die mit „−“ bezeichnet wird (also beispielsweise HFE$^-$), besteht darin, die Einwegfunktion P nicht vollständig zu veröffentlichen. Das bedeutet, statt $M(P) = (p_1, \ldots, p_n)$ zu veröffentlichen, werden die hinteren k Polynome $p_{n-k+1}, \ldots, p_n$ geheimgehalten und nur die ersten $n - k$ Polynome $P_- := (p_1, \ldots, p_{n-k})$ veröffentlicht. Das Vorgehen beim Entschlüsseln und beim Signieren muss dann entsprechend angepasst werden:

- Da dem Verschlüsseler nur die ersten $n - k$ Polynome bekannt sind, kann er dem Empfänger nur $P_-(x) = (y_1, \ldots, y_{n-k})$ schicken. Der Empfänger muss also nun seinerseits alle q^k möglichen Endungen $(y_{n-k+1}, \ldots, y_n)$ durchprobieren und alle dazugehörigen Lösungen x berechnen. Aufgrund der Redundanz ist er dann hoffentlich (vgl. Abschnitt 2.3.3) in der Lage, das richtige x zu bestimmen.
  Soll HFE$^-$ zum Verschlüsseln angewendet werden, ist es also wichtig, dass nicht zu viele Polynome geheimgehalten werden, da dadurch sowohl die Effizienz als auch

die Sicherheit, eindeutig entschlüsseln zu können, beeinträchtigt wird. Um letzteres auszugleichen, wäre es also sinnvoll, ein wenig mehr Redundanz — wie in Abschnitt 2.2 beschrieben — einzufügen.

- Soll HFE^- als Signatur-Schema eingesetzt werden, kann k deutlich größer gewählt werden. Denn ähnlich wie bei der in Abschnitt 2.2 vorgestellten 160-Bit-Signatur, können die geheimgehaltenen Stellen sozusagen als Randomisierung dienen, und sind so für den Signierer sogar von Vorteil, da es leichter ist, eine „signierbare“ Nachricht zu finden. Es muss lediglich darauf geachtet werden, k nicht so groß zu wählen, dass das Problem, zu gegebenen $(y_1, \ldots, y_{n-k})$ (ohne Kenntnis des geheimen Schlüssels) eine passende Signatur x zu finden, zu einfach wird (siehe Kapitel 4.1). In [CGP98b] schlägt Patarin beispielsweise Werte von $k = 1, 2$ oder $\frac{n}{2}$ als praktisch und effizient vor.

2.4.2 Hinzufügen von zufälligen Polynomen („ + “)

Genau im Gegensatz zur ersten Idee steht die zweite Perturbation „+“. Dabei werden, statt Polynome wegzulassen, noch k weitere, zufällig gewählte Polynome hinzugefügt. Natürlich machte es keinen Sinn, diese zusätzlichen Polynome einfach an $(p_1, \ldots, p_n)$ hinten anzuhängen, denn dann könnte ein Angreifer leicht das ursprüngliche P zurückerhalten. Stattdessen werden die Polynome schon zur multivariaten Darstellung von φ_{K^n} hinzugefügt und dann durch die Transformationen s und t mit den übrigen Polynomen vermischt. So kann als öffentliche Funktion eine Abbildung $P_+ : K^n \to K^{n+k}$ verwendet werden, dargestellt durch $M(P_+) = (p_1, \ldots, p_{n+k})$, aus der sich ohne Kenntnis von s und t die zusätzlichen Polynome von den übrigen nicht mehr trennen lassen.
Auch die Wahl von k steht bei „+“ genau im Gegensatz zur Wahl bei „−“:

- Für die k zusätzlichen, zufällig gewählten Polynome gibt es keine Möglichkeit sie effizient umzukehren. Die Wahrscheinlichkeit, dass eine Lösung der ursprünglichen Gleichungen auch die durch die zusätzlichen Polynome gegebenen Gleichungen erfüllt, beträgt nur $\frac{1}{q^k}$. Es müssen also durchschnittlich q^k Lösungen für das ursprüngliche System berechnet werden, bis eine Lösung des Gesamtsystems gefunden wird.
  Damit das Signieren nicht zu ineffizient wird, muss k deshalb in einem auf HFE^+ beruhenden Signatur-Schema ähnlich wie beim Entschlüsseln mit HFE^-, ziemlich klein gewählt werden.

- Soll HFE^+ dagegen zum Verschlüsseln eingesetzt werden, kann k prinzipiell wesentlich größer gewählt werden. Die k zusätzlichen Polynome können beim eigentlichen Entschlüsseln ignoriert werden und bieten sogar noch eine gewisse Redundanz wie schon in Abschnitt 2.2 angedeutet wurde, so dass die zusätzlich einzufügende

Redundanz kleiner gewählt werden kann.

Zu beachten ist dabei allerdings, dass k nicht so groß gewählt wird, dass das Entschlüsseln auch ohne Kenntnis des geheimen Schlüssels zu einfach wird. Auf keinen Fall darf k beispielsweise größer als $\frac{n(n+1)}{2} - n$ gewählt werden, da das System sonst mittels einer Linearisierung (vgl. Kapitel 4.2.1) effizient lösbar ist. Aber auch kleinere k können unter Umständen schon das Lösen solcher Systeme in subexponentieller Zeit erlauben, wie in Abschnitt 4.2.3 gezeigt wird, weshalb k sorgfältig gewählt werden sollte (vgl. auch Abschnitt 4.3).

2.4.3 Hinzufügen von Variablen („ V “)

Anstatt die Anzahl der verwendeten Polynome zu variieren, ist es auch möglich, die Anzahl der verwendeten Variablen zu erhöhen. Dieses im Folgenden genauer beschriebene Vorgehen wird mit „V“ bezeichnet.

Dabei müssen die zusätzlichen Variablen $(x_{n+1}, \ldots, x_{n+k})$ auch wieder gut versteckt und mit den anderen Variablen vermischt werden, damit ein Angreifer das ursprüngliche Schema nicht leicht erkennen oder sogar zurückerhalten kann. Patarin schlägt beispielsweise folgendes Vorgehen vor, das an sein eigenständiges Kryptosystem UOV (Unbalanced Oil & Vinegar, siehe dazu auch [CGP98b] und [GKP99]) angelehnt ist :

Das HFE-Polynom $f(x) = \sum_{i,j} \beta_{ij} x^{q^i+q^j} + \sum_i \alpha_i x^{q^i} + \mu_0$ wird dabei nicht fest, sondern in Abhängigkeit von den neuen Variablen $x_{n+1}, \ldots, x_{n+k}$ gewählt, indem die Konstanten $\alpha_0, \ldots, \alpha_{n-1}, \mu_0 \in L$ durch Funktionen $K^k \to L$ ersetzt werden.. Damit die Darstellung letztendlich wieder von der gewünschten quadratischen Form ist, werden dazu die Koeffizienten α_i durch lineare Funktionen $\alpha_i : K^k \to L$ und μ_0 durch eine quadratische Funktion $\mu_0 : K^k \to L$ ersetzt. Dies liefert eine Funktion $\widetilde{\varphi} : K^{n+k} \to K^n$, die anstelle der Funktion φ_{K^n} im ursprünglichen Schema verwendet werden kann. Durch in der Größe angepasste Transformationen s und t werden dann die so hinzugefügten Variablen mit den ursprünglichen Variablen vermischt und versteckt.

- Diese Variante ist wiederum ähnlich wie „−“ gut als Signaturverfahren geeignet. Um eine passende Signatur zu finden, können nach Rücktransformation mittels s und t, die zusätzlichen Variablen $x_{n+1}, \ldots, x_{n+k}$ einfach mit beliebigen, zufälligen Werten belegt werden, wonach eine Signatur wie üblich bestimmt werden kann. Die zusätzlichen Variablen bieten also ebenfalls eine Möglichkeit, die benötigte Randomisierung einzubauen.

- Zum Entschlüsseln ist dieses Verfahren schlechter geeignet bzw. wiederum nur für kleine Werte von k durchführbar, da wie auch schon bei „−“ alle möglichen Belegungen für die zusätzlichen Variablen ausprobiert werden müssen. Dies senkt aber die Effizienz und macht zudem eine größere Redundanz erforderlich.

Eine ausführlichere Beschreibung und Analyse von Verfahren, die auf dem Hinzufügen und Mischen von Variablen beruhen, gibt Patarin in [GKP99].

2.4.4 Fixieren von Variablen („F“)

Wie schon die Anzahl der veröffentlichten Polynome erhöht („+“) und vermindert („−“) werden konnte, so können auch die verwendeten Variablen nicht nur vermehrt („V“), sondern auch vermindert werden. Diese Idee, das Verfahren abzuändern, führt zum sogenannten Fixieren einiger Variablen, das mit „F“ bezeichnet wird. Dabei werden k der Variablen in der Darstellung der eigentlich öffentlichen Funktion P (etwa die Variablen $x_{n-k+1}, \ldots, x_n$) auf zufällige, feste Werte gesetzt und die so erhaltene Funktion $P_F(x_1, \ldots, x_{n-k})$ wird veröffentlicht.

- Zum Entschlüsseln ist dieses Verfahren wiederum auch für etwas größere k gut geeignet, da bei der Entschlüsselung die festen, also schon bekannten x_i-Werte als zusätzliche Redundanz betrachtet werden können, um falsche x auszuschließen. Wie auch schon bei „+“ muss dabei nur beachtet werden, dass k nicht so groß gewählt wird, dass das Gleichungssystem direkt zu lösen ist (vgl. Kapitel 4).
- Als Signaturverfahren ist diese Variante dagegen wiederum nur für kleine k geeignet, da jede fixierte Variable die Wahrscheinlichkeit, dass zu einem gegebenen y ein passendes x existiert um den Faktor $\frac{1}{q}$ senkt.

2.4.5 Zusammenfassung

In folgender Tabelle sind noch einmal die wesentlichen Eigenschaften der vier Perturbationen veranschaulicht:

	Variation in der Anzahl der			
	Polynome		Variablen	
	-	+	V	F
bei der Verschlüsselung				
nötige zusätzl.Redundanz	größer	kleiner	größer	kleiner
mögliche Wahl von k	eher klein	eher groß	eher klein	eher groß
als Signatur				
nötige zusätzl.Random.	kleiner	größer	kleiner	größer
mögliche Wahl von k	eher groß	eher klein	eher groß	eher klein
besser geeignet für	Signatur	Verschlüsselung	Signatur	Verschlüsselung

Diese Übersicht zeigt deutlich, dass die hier vorgestellten Variationen in zwei recht gegensätzliche Gruppen einzuteilen sind.

„−“ und „V“ vergrößern das Verhältnis der Größe des Urbildraumes der öffentlichen Funktion P zur Größe ihres Zielraumes von $\frac{q^n}{q^n} = 1$ zu $\frac{q^n}{q^{n-k}} = \frac{q^{n+k}}{q^n} = q^k$. Die Chance, dass P schon „annähernd surjektiv“ ist, ist also viel größer und bei einer Verwendung als Signaturverfahren sind somit weniger Randomisierungen zusätzlich einzufügen. Diese beiden Perturbationen bringen also eher für Signaturverfahren Vorteile und sind auch für große k noch gut zu gebrauchen.
Die Perturbationen „+“ und „F“ dagegen verkleinern das beschriebene Verhältnis dagegen auf $\frac{q^n}{q^{n+k}} = \frac{q^{n-k}}{q^n} = q^{-k}$. Die hierzu gehörigen Einwegfunktionen P haben also eine viel größere Wahrscheinlichkeit „annähernd injektiv“ zu sein. Bei der Verwendung als Verschlüsselungsverfahren muss damit viel weniger zusätzliche Redundanz eingefügt werden, um eine eindeutige Entschlüsselung zu ermöglichen, so dass diese beiden Perturbationen eher für Verschlüsselungsverfahren geeignet sind.

Eine offene Frage ist bislang, ob diese Variationen die Sicherheit der Verfahren wirklich wie erhofft steigern oder ob die Sicherheit durch solche Veränderungen sogar geschwächt wird. Auf jeden Fall wird die ursprünglich rein algebraische Natur des HFE-Schemas durch solche Verfahren zerstört und es entstehen Verfahren von eher kombinatorischer Natur.

Die Perturbationen können kombiniert werden, wobei sich natürlich vor allem die Kombinationen „V^-“ und „F^+“ anbieten, aber auch andere Kombinationen wie etwa „$+-$“ oder „VF^{+-}“ können durchaus sinnvoll sein.
Wie schon zu Anfang dieses Abschnittes erwähnt, können diese Perturbationen nicht nur auf HFE angewendet werden, sondern auch auf andere, ähnlich aufgebaute Schemata. So kann beispielsweise das in Abschnitt 2.1 vorgestellte C^* auch durch Anwendung von „$-$“ repariert werden, wie Patarin in [CGP98b] zeigt. Allerdings muss dazu k sehr groß gewählt werden, so dass sich das entstehende Verfahren, das aufgrund des großen k mit C^{*--} bezeichnet wird, nur noch als Signaturschema eignet. Ebenfalls in [CGP98b] wird C^{*-+} beschrieben, eine weitere C^*-Variante, die ebenfalls, für den Fall, dass genügend Polynome geheimgehalten werden, sicherer erscheint als C^*.

In Kapitel 4 wird noch einmal darauf eingegangen, wie groß k bei einigen dieser Perturbationen gewählt werden darf, damit das Verfahren nicht direkt gebrochen werden kann.

2.5 Sicherheitsgrundlagen

Die Sicherheit der meisten heutzutage verwendeten Kryptoverfahren, wie RSA oder Verfahren, die mit elliptischen Kurven arbeiten, steht in engem Zusammenhang zu bekannten zahlentheoretischen Problemen, wie der Faktorisierung großer Zahlen oder dem Berechnen

diskreter Logarithmen. Beispielsweise ist klar, dass jemand der effizient große Zahlen faktorisieren kann, auch RSA brechen kann.
Die Sicherheit von HFE dagegen steht nicht in Zusammenhang zu diesen beiden „Standardproblemen“. Dies kann sowohl als Vorteil als auch als Nachteil aufgefasst werden.
Von Vorteil ist, dass sich an der Sicherheit des HFE-Systems nichts änderte, wenn es auf einmal möglich wäre, effizient zu faktorisieren bzw. diskrete Logarithmen zu berechnen. Fast alle anderen Public-Key-Kryptosysteme dagegen beruhen direkt auf der Schwierigkeit eines dieser beiden Probleme. Diese Systeme brächen also alle zusammen, würden diese beiden Probleme gelöst. Dies ist zwar sehr unwahrscheinlich, es sollte aber dennoch in Betracht gezogen werden.
Andererseits ist von Nachteil, dass die Sicherheit von HFE damit nur auf anderen, unter Umständen nicht so gut studierten Problemen begründet werden kann. Dies hat natürlich zur Folge, dass das Vertrauen in die Sicherheit deutlich geringer ist, als bei anderen, bekannten Verfahren, so dass es wichtig ist, zu untersuchen, worauf die Sicherheit von HFE eigentlich begründet ist. Dies soll in den folgenden Abschnitten geschehen.

2.5.1 Zugrunde liegende Probleme

In diesem Abschnitt werden die grundlegenden Probleme beschrieben, die mit der Sicherheit von HFE in Beziehung stehen.
Courtois hat in [Cou01a] versucht, eine Definition dafür zu finden, was es bedeutet, das HFE-Kryptosystem zu brechen. Er bezeichnet das dabei zu lösende Problem einfach als HFE-Problem (HFEP):

Problem 2.5.1 (HFEP)
Seien $y \in K^n$ und eine Funktion $P : K^n \to K^n$ gegeben, die, wie in Abschnitt 2.1 beschrieben, mit einer zu einem HFE-Polynom $f \in L[x]$ gehörigen Funktion $\varphi : L \to L$ und zwei bijektiven, affinen Transformationen $s, t : K^n \to K^n$ als

$$P = t \circ \varphi_{K^n} \circ s$$

geschrieben werden kann.
*Dann besteht das **HFE-Problem** (kurz **HFEP**) darin, ohne Kenntnis der Zerlegung von P in $t \circ \varphi_{K^n} \circ s$ ein $x \in K^n$ mit $P(x) = y$ zu finden.*

Das von Courtois definierte HFEP modelliert nicht das globale Brechen eines HFE-Systems, also das Finden des geheimen Schlüssels bzw. ähnlich nützlicher Informationen, mit deren Hilfe zu beliebigen y die Lösungen x bestimmt werden können, sondern nur das Brechen bezüglich eines festen $y \in K^n$.
Da der öffentliche Schlüssel P in Form von quadratischen Polynomen gegeben ist, kann $P(x) = y$ für festes y im Wesentlichen als quadratisches Gleichungssystem in den x-Variablen aufgefasst werden. Das HFEP hängt also offensichtlich eng mit dem Lösen

multivariater, quadratischer Gleichungssysteme, dem sogenannten Problem MQ, zusammen:

Problem 2.5.2 (MQ)
Gegeben sei ein beliebiger Körper K und ein System

$$\sum_{1 \le i \le j \le n} \mu_{ijk} x_i x_j + \sum_{i=1}^{n} \nu_{ik} x_i = \eta_k \qquad \text{für } k = 1, \ldots, m$$

*von m quadratischen Gleichungen in n Unbekannten $x_1, \ldots, x_n$ mit μ_{ijk}, ν_{ik}, $\eta_k \in K$. Dann besteht das Problem **MQ** (MQ=**m**ultivariat **q**uadratisch) darin, (mindestens) eine Lösung $(a_1, \ldots, a_n) \in K^n$ dieses Systems zu finden.*

Leider beschreibt dieses relativ allgemeine und bekannte algebraische Problem nicht genau das im Falle von HFE zu lösende Problem. Natürlich ist das HFEP auf MQ reduzierbar, d.h. jemand, der MQ effizient lösen kann, ist auch in der Lage HFE zu brechen. Die umgekehrte Richtung, die wichtig wäre, um die Sicherheit von HFE relativ zu MQ zu beweisen, gilt aber leider nicht.
Dies liegt daran, dass das bei HFE-Systemen gegebene, spezielle MQ-Problem immer so aufgebaut ist, dass es unter Kenntnis des geheimen Schlüssels effizient umkehrbar ist. In Abschnitt 2.3.2 wurde aber gezeigt, dass dafür der Grad d des zugrunde liegenden HFE-Polynoms nicht zu groß gewählt werden darf. Andererseits können, wie in Abschnitt 2.3.1 gezeigt wurde, beliebige Systeme von quadratischen Gleichungssystemen wie sie im allgemeinen Problem MQ vorkommen, durch HFE-Polynome nur erzeugt werden, wenn für diese ein Grad von $d = 2 \cdot q^{n-1}$ zugelassen wird.
HFEP ist also nur ein Spezialfall von MQ, ein Beweis der Schwierigkeit von MQ beweist also leider nichts über die Sicherheit von HFE. Allerdings ist es dennoch sinnvoll, MQ näher zu untersuchen: Zum einen würde eine effiziente Lösung von MQ auch eine effiziente Lösung für HFEP, also das Brechen von HFE, implizieren, und zum anderen deutet die Schwierigkeit, eine solche Lösung zu finden, zumindest an, dass es auch schwierig sein könnte, HFE zu brechen.

Auch wenn — wie oben bemerkt — das HFEP ausdrücklich nicht das globale Brechen, also das Finden des geheimen Schlüssels beschreibt, ist dies natürlich dennoch eine weitere Möglichkeit, das HFEP zu lösen. Auch dieser Ansatz sollte somit näher untersucht werden, um mehr über die Sicherheit von HFE zu erfahren.

Das Finden des geheimen Schlüssels, also der affinen Transformationen s und t bzw. zweier ähnlich hilfreicher Transformationen, wird durch zwei verschiedene Problemstellungen beschrieben, abhängig davon, ob dem Angreifer die versteckte Funktion φ bekannt ist oder nicht.

Ist φ bekannt, so ist auch ein φ_{K^n} für eine beliebige Darstellung $L \cong K^n$ bekannt, und das Problem, die beiden affinen Transformationen s und t zu finden, wird genau durch das Problem „IP mit zwei Geheimnissen“ (siehe auch [Pat96a] und [CGP98a]) beschrieben:

Problem 2.5.3 (IP)
Sei A eine Menge von m Gleichungen der Form

$$y_k = \sum_{i,j} \gamma_{ijk} x_i x_j + \sum_i \mu_{ik} x_i + \delta_k \qquad \text{für } k = 1, \ldots, m$$

in den Variablen $x_1, \ldots, x_n$ und $y_1, \ldots, y_m$ mit $\gamma_{ijk}, \mu_{ik}, \delta_k \in K$. Seien weiter s und t zwei bijektive, affine Transformationen der Variablen mit $s(x_1, \ldots, x_n) = (x'_1, \ldots, x'_n)$ und $t(y_1, \ldots, y_m) = (y'_1, \ldots, y'_m)$ und B die Menge der sich daraus ergebenden Gleichungen der Form

$$y'_k = \sum_{i,j} \gamma'_{ijk} x'_i x'_j + \sum_i \mu'_{ik} x'_i + \delta'_k \qquad \text{für } k = 1, \ldots, m.$$

*Dann besteht das Problem **IP** (=„**I**somorphism of **P**olynomials“) darin, zu gegebenen Mengen A und B der obigen Form zwei bijektive, affine Transformationen s und t zu finden, die A wie oben beschrieben in B überführen.*

Als Mengen A und B können im Fall von HFE die Gleichungen gewählt werden, die die Beziehungen $P(x) = y$ und $\varphi_{K^n}(x') = y'$ beschreiben und aus den multivariaten Darstellungen $M(P)$ und $M\left(\varphi_{K^n}\right)$ entstehen. Wäre φ öffentlich bekannt, lieferte eine effiziente Lösung von IP also direkt den geheimen Schlüssel und das Verfahren wäre gebrochen.

Ist dem Angreifer φ dagegen unbekannt, so weiss er aufgrund der Konstruktion des HFE-Verfahrens zumindest, dass unter all den Abbildungen $\varphi = \phi^{-1} \circ t^{-1} \circ P \circ s^{-1} \circ \phi$ für verschiedene Transformationen s und t mindestens ein φ existiert, für das das zugehörige HFE-Polynom $U(\varphi)$ einen besonders kleinen Grad hat.
In Abschnitt 5.2 wird eine Attacke von Aviad Kipnis und Adi Shamir beschrieben, die dieses Wissen ausnutzt und auf Aussagen über bestimmte Matrizen überträgt. Das Finden der Transformationen s und t wird so auf eine spezielle Form des im Folgenden definierten Problems MinRank gebracht.
Der genaue Zusammenhang von MinRank und HFE wird in Abschnitt 5.2 dargestellt.

Problem 2.5.4 (MinRank)
Gegeben seien m quadratische $n \times n$-Matrizen $M_1, \ldots, M_m$ über einem Körper K und eine natürliche Zahl $r < n$.
*Dann besteht das Problem **MinRank** darin, $\alpha_1, \ldots, \alpha_m \in K$ zu finden, so dass*

$$Rang\left(\sum_{i=1}^{m} \alpha_i M_i\right) \quad \leq \quad r$$

gilt.

Das HFE-Problem kann also als Spezialfall der drei Probleme MQ, IP und MinRank betrachtet werden. Im folgenden Abschnitt wird nun beschrieben, wie die Schwierigkeit dieser Probleme aus komplexitätstheoretischer Sicht zu bewerten ist.

2.5.2 Komplexitätstheoretische Betrachtungen

Zu den folgenden Bemerkungen aus dem Bereich der Komplexitätstheorie ist zunächst Folgendes anzumerken (vgl. dazu auch Bemerkung 1.3.25): Aussagen darüber, dass ein Problem $\mathcal{NP}$-hart ist, sind immer Aussagen über das worst case Verhalten dieses Problems. Im average case, der in der Kryptographie wesentlich relevanter ist, können $\mathcal{NP}$-harte Probleme durchaus deutlich einfacher, beispielsweise in polynomieller Zeit lösbar, sein. Dennoch macht es Sinn, solche Aussagen bezüglich der oben genannten Probleme zu betrachten, um zu sehen, dass die Probleme zumindest in voller Allgemeinheit nicht einfach zu lösen sind.
Aus komplexitätstheoretischer Sicht sind die drei oben definierten Probleme MQ, IP und MinRank alle schwierig. Genauer gesagt sind MQ und MinRank $\mathcal{NP}$-hart und auch IP scheint zumindest nicht in $\mathcal{P}$ zu liegen. Die Schwierigkeit soll nun zunächst am Beispiel von MQ genauer gezeigt werden:

Satz 2.5.5
MQ ist $\mathcal{NP}$-hart.
Beweis:
Es genügt zu zeigen, dass die zu MQ gehörige Entscheidungsvariante (im Folgenden mit MQE bezeichnet), die danach fragt, ob das gegebene Gleichungssystem lösbar ist, $\mathcal{NP}$-hart ist. Dazu wird zunächst der Fall $K = \mathbb{F}_2$ betrachtet und das bekanntermaßen $\mathcal{NP}$-vollständige Problem 3-SAT (siehe Problem 1.3.17 und Satz 1.3.18) auf die Entscheidungsvariante MQE reduziert:
Sei dazu (U, C) eine Instanz des 3-SAT-Problems, wobei $U = \{u_1, \ldots, u_n\}$ die Menge der Variablen und $C = \{c_1, \ldots, c_m\}$ die Menge der Klauseln bezeichne. Um 3-SAT nun auf MQE zu reduzieren, ist es nötig, diese Instanz in ein Gleichungssystem über $\mathbb{F}_2$ zu überführen, dass genau dann eine Lösung besitzt, wenn es für U eine Belegung gibt, die die Klauseln C erfüllt.
Die Belegung der Variablen $u_1, \ldots, u_n$ kann in $\mathbb{F}_2$ dargestellt werden, mit 1 für $wahr$ und 0 für $falsch$. Wird u_i dabei durch die Variable x_i repräsentiert, so entspricht in dieser Schreibweise das Literal $\overline{u_i}$ dem Ausdruck $1 - x_i$.
Jede Klausel c_i ist eine Disjunktion von 3 Literalen, also von der Form $\widetilde{u_1} \vee \widetilde{u_2} \vee \widetilde{u_3}$ mit $\widetilde{u_i} \in \{u_1, \ldots, u_n, \overline{u_1}, \ldots, \overline{u_n}\}$. Durch Einsetzen der möglichen Werte kann leicht überprüft werden, dass eine solche Klausel genau dann erfüllt werden kann, wenn die Gleichung

$$\widetilde{x_1} + \widetilde{x_2} + \widetilde{x_3} + \widetilde{x_1}\widetilde{x_2} + \widetilde{x_1}\widetilde{x_3} + \widetilde{x_2}\widetilde{x_3} + \widetilde{x_1}\widetilde{x_2}\widetilde{x_3} = 1 \qquad \text{mit } \widetilde{x_i} = \begin{cases} x_j & \text{für } \widetilde{u_i} = u_j, \\ 1 - x_j & \text{für } \widetilde{u_i} = \overline{u_j} \end{cases}$$

eine Lösung in $\mathbb{F}_2$ hat. Auf diese Weise können alle Klauseln in Gleichungen übertragen werden. So kann also insgesamt die vorgegebene Instanz von 3-SAT in ein kubisches Gleichungssystem über $\mathbb{F}_2$ transformiert werden, das genau dann eine Lösung hat, wenn es eine Belegung für die u_i gibt, die alle Klauseln gleichzeitig erfüllt.
Dieses kubische Gleichungssystem wiederum kann durch Substitution der Variablen $x_i x_j =: z_{ij}$ und Hinzufügen der Gleichungen $z_{ij} = x_i x_j$ in ein quadratisches Gleichungssystem verwandelt werden, das genau dann lösbar ist, wenn es schon das ursprüngliche, kubische Gleichungssystem war. Dieses letzte Gleichungssystem ist aber nun eine gültige Eingabe für MQE, deren Größe zudem noch polynomiell bezüglich der Größe von (U, C) beschränkt ist. Damit ist 3-SAT also polynomiell auf MQE für den Fall $K = \mathbb{F}_2$ reduzierbar, d.h. MQE für $K = \mathbb{F}_2$ ist $\mathcal{NP}$-hart.

Um nun zum allgemeinen Fall mit beliebigem K zu gelangen, muss noch das MQE-Problem über $\mathbb{F}_2$ auf MQE über beliebigem K reduziert werden. Dies geschieht wiederum, indem ein beliebiges quadratisches Gleichungssystem über $\mathbb{F}_2$ in ein Gleichungssystem über K transformiert wird, das genau dann lösbar ist, wenn schon das Ausgangssystem lösbar ist. Die Idee dabei ist, $\mathbb{F}_2$ in gewisser Weise in K „einzubetten“, natürlich unter Anpassung der Verknüpfungen. Die Multiplikation in K kann dabei erhalten bleiben, da in jedem Körper K für $a, b \in \{0, 1\}$ das Produkt $a \cdot b$ das gleiche Ergebnis aus $\{0, 1\}$ liefert. Die Addition $(x, y) \mapsto x+y$ in $\mathbb{F}_2$ muss dagegen ersetzt werden durch die Abbildung $(x, y) \mapsto (x + y)(2 - (x + y))$ in K, die sicherstellt das die Summen jeweils wieder aus $\{0, 1\} \subseteq K$ sind.
Zusammen mit hinzuzufügenden Gleichungen der Form $x(x - 1) = 0$, die dafür sorgen, dass die Lösungen nur aus $\{0, 1\} \subseteq K$ stammen können, liefert dieses Vorgehen das gewünschte Gleichungssystem über K. Details dieser Transformation können im Anhang von [GP97b] nachgelesen werden. ■

Dieser Satz zeigt also, dass MQ zumindest im worst case schwierig ist. Wie schwierig MQ im average case ist, ist damit natürlich nicht klar. Einige Ergebnisse über spezielle für HFE interessante Spezialfälle von MQ werden später in den Kapiteln 3 und 4 vorgestellt.

Für das MinRank-Problem gilt ähnliches wie für MQ. In [BFS96] wird beispielsweise gezeigt, dass die Entscheidungsvariante für den Fall endlicher Körper $\mathcal{NP}$-vollständig ist, so dass auch das allgemeine MinRank-Problem für endliche Körper $\mathcal{NP}$-hart ist. Für bestimmte r, die deutlich kleiner als n sind, scheint dagegen auch dieses Problem effizienter lösbar zu sein, wie beispielsweise in den Spezialfällen, die in der Attacke in Abschnitt 5.2 zu lösen sind.

Im Gegensatz dazu kann für IP gezeigt werden, dass die zugehörige Entscheidungsvariante vermutlich nicht $\mathcal{NP}$-hart ist (siehe [CGP98a]). Dort wird aber auch gezeigt, dass IP

mindestens so schwierig ist wie das Graph-Isomorphismus-Problem, von dem allgemein vermutet wird, dass es nicht in polynomieller Zeit gelöst werden kann. Auch das Problem IP scheint also — wenn auch nicht $\mathcal{NP}$-hart — so doch zumindest schwierig, d.h. nicht in polynomieller Zeit lösbar, zu sein.
Die bislang besten bekannten Algorithmen (siehe [CGP98a]) lösen IP für lineare Transformationen s und t in $O\left(q^{\frac{n}{2}}\right)$ und für affine Transformationen in $O\left(q^{\frac{3}{2}n}\right)$. Allerdings hat Patarin in einer E-Mail bestätigt, dass vermutlich auch IP mit affinen Transformationen in $O\left(q^{\frac{n}{2}}\right)$ zu lösen ist.
Wenn die versteckte Funktion φ veröffentlicht werden soll (wie in Abschnitt 2.2 beschrieben) müsste n also doppelt so groß wie sonst üblich gewählt werden, um Attacken über IP auszuschließen.

In Bezug auf das eigentlich zu untersuchende Problem HFEP sind Aussagen wie beispielsweise Satz 2.5.5 ziemlich schwach, da HFEP nur ein Spezialfall von MQ ist, also selbst nicht $\mathcal{NP}$-hart sein muss, nur weil MQ $\mathcal{NP}$-hart ist.
Es ist allerdings auch unwahrscheinlich, dass es möglich ist, zu zeigen, dass HFEP selbst $\mathcal{NP}$-hart ist, wie der folgende Satz zeigt. Der Beweis läuft analog zur Argumentation von Brassard in [Bra79], mit der er eine ähnliche Aussage für das discrete log problem beweist.

Satz 2.5.6
Wenn $\mathcal{NP} \neq$ co-$\mathcal{NP}$ gilt, dann ist HFEP nicht $\mathcal{NP}$-hart.

Bemerkung 2.5.7
$\mathcal{NP} \neq$ co-$\mathcal{NP}$ ist eine allgemein verbreitete Annahme in der Komplexitätstheorie (vgl. Abschnitt 1.3.2).
Beweis (von Satz 2.5.6):
Seien P und $y \in K^n$ entsprechend dem HFEP gegeben. Dann können mit einer nichtdeterministischen Turingmaschine folgendermaßen alle $x \in K^n$ mit $P(x) = y$ bestimmt werden:
Rate zwei bijektive, affine Transformationen s und t und einen Isomorphismus $\phi : L \to K^n$ und transformiere damit das über K^n durch $P(x) = y$ gegebene Gleichungssystem wie üblich in die univariate Gleichung $f(a) = b$ über L. Durch den Nichtdeterminismus kann dabei gesichert werden, dass zwei Transformationen s und t geraten werden, so dass der Grad des erhaltenen f klein ist. Nun kann noch zusätzlich eine Faktorisierung von f geraten werden, aus der alle Lösungen x bestimmt werden können.
Betrachte nun folgendes zu HFEP gehörige Entscheidungsproblem, das auch als Projektion $\mathrm{P}_{\mathrm{HFEP}}$ bezeichnet wird:

Seien P und y wie in HFEP, $z \in K^n$ und eine beliebige totale Ordnung $<$ auf K^n gegeben. Existiert ein $x \in K^n$ mit $P(x) = y$ und $x < z$?

Mit Hilfe des oben gegebenen Algorithmus ist dann leicht zu sehen, dass $\mathrm{P}_{\mathrm{HFEP}}$ sowohl in $\mathcal{NP}$ als auch in co-$\mathcal{NP}$ liegt, also $P_{HFEP} \in \mathcal{NP} \cap \text{co-}\mathcal{NP}$ gilt.
Nach Satz 1.3.24 folgt wegen der Annahme $\mathcal{NP} \neq \text{co-}\mathcal{NP}$ damit aber, dass $\mathrm{P}_{\mathrm{HFEP}}$ nicht $\mathcal{NP}$-hart sein kann. Weiter ist HFEP aber mit einem $\mathrm{P}_{\mathrm{HFEP}}$-Orakel mit Hilfe eines divide-and-conquer-Ansatzes in polynomieller Zeit lösbar. Wäre HFEP $\mathcal{NP}$-hart, wäre also auch $\mathrm{P}_{\mathrm{HFEP}}$ $\mathcal{NP}$-hart im Widerspruch zur obigen Feststellung. ■

Dieser Satz sagt also aus, dass es vermutlich nicht möglich ist, zu zeigen, dass das HFEP $\mathcal{NP}$-hart, d.h. im komplexitätstheoretischen Sinne schwierig ist. Dies gilt allerdings nicht nur für HFE, sondern auch für die meisten anderen auf Einwegfunktionen basierenden Kryptosysteme. Denn Brassard schreibt in [Bra79] weiter, dass sich eine dem obigen Satz entsprechende Aussage für alle Einwegfunktionen, die einige wenige Bedingungen erfüllen, auf diese Weise treffen lässt. Im Allgemeinen ist es für ein auf einer Einwegfunktion basierendes Public-Key-Kryptosystem also vermutlich nicht möglich, zu beweisen, dass das Brechen dieses Systems $\mathcal{NP}$-hart ist.

2.5.3 Überblick

Da sich die Sicherheit von HFE nicht beweisen lässt (bislang auch nicht relativ zu einem bekannten Problem), argumentiert Patarin in [Pat96a] folgendermaßen, warum HFE trotzdem vermutlich sicher ist:
Mit steigendem Grad d des versteckten Polynoms f können immer mehr öffentliche Funktionen P beschrieben werden. Insbesondere kann, wie in Abschnitt 1.3.1 gezeigt, für maximales $d = 2q^{n-1}$ jede beliebige Funktion P mit $M(P) \in \mathcal{M}_{n,2}$ auf diese Weise erzeugt werden. Patarin argumentiert nun, dies bedeute, dass die Eigenschaften, die HFE-Funktionen P von zufällig gewählten, multivariaten, quadratischen Funktionen P (d.h. mit $M(P) \in \mathcal{M}_{n,2}$) unterscheiden, mit wachsendem d immer mehr verschwinden. Das Problem dabei sei nur, dass d nicht maximal gewählt werden könne, damit das Verfahren effizient bleibe.

Das Problem an dieser Argumentation ist, dass weniger die Effizienz an sich als vielmehr das allgemeine Prinzip eines Public-Key-Verfahrens erforderlich macht, d viel kleiner als maximal möglich zu wählen. Denn Ziel eines guten Public-Key-Systems sollte es immer sein, die Differenz zwischen der Zeit, in der die Einwegfunktion **ohne Kenntnis** des geheimen Schlüssels umgekehrt werden kann, und der Zeit, in der dies **mit Kenntnis** des geheimen Schlüssels geschehen kann, zu maximieren. Denn nur, wenn diese Spanne ziemlich groß ist, ist es möglich, die Parameter des Verfahrens so zu wählen, dass es einerseits so effizient ist, dass es beispielsweise auch auf Smartcards gut ausgeführt werden kann, aber andererseits dennoch so sicher ist, dass es auch mit Großrechenanlagen nicht gebrochen werden kann.
Bei HFE wird eine obere Schranke für diese Spanne aber gerade durch das Verhältnis

des Grades d des versteckten Polynoms zum maximal möglichen Grad $2 \cdot q^{n-1}$ vorgegeben: In Abschnitt 2.3.2 wurde gezeigt, dass der legitime Benutzer des Verfahrens, der den geheimen Schlüssel kennt, zur Umkehrung im Wesentlichen eine Zeit von $O\left(d\left(\log\left(d\right)\right)^{O(1)} \cdot \left(dn^2 + mn^3\right)\right)$ benötigt, die sich aus dem Suchen der Nullstellen des versteckten Polynoms f vom Grad d ergibt. Aber auch ein Angreifer, der die geheimen Schlüssel nicht kennt, kann, wie aus den Sätzen aus Abschnitt 1.3.1 hervorgeht, durch Wahl beliebiger Transformationen s und t zumindest ein univariates Polynom f' von einem Grad $\leq 2 \cdot q^{n-1}$ erhalten, dessen Nullstellen ebenfalls die gesuchten Lösungen liefern. Die dafür benötigte Zeit unterscheidet sich aber wegen der unterschiedlichen Grade der Polynome im Wesentlichen durch einen Faktor von $\left(\frac{2 \cdot q^{n-1}}{d}\right)^2$.

In den folgenden Kapiteln wird nun untersucht, wie effizient die anderen Möglichkeiten sind, das HFE-Problem zu lösen. Dabei wird vor allem die Schwierigkeit von MQ näher betrachtet.
In Kapitel 3 wird zunächst allgemein beschrieben, wie Gröbnerbasis-Techniken zum Lösen von Gleichungssystemen auf das Problem MQ angewendet werden können. Insbesondere wird beschrieben, welche Vorteile der bei HFE auftretende Spezialfall von MQ bei der Anwendung dieser Techniken bringt.
Im darauffolgenden Kapitel 4 werden dann Algorithmen für spezielle Fälle von MQ, genauer für überbestimmte und unterbestimmte Gleichungssysteme beschrieben. Diese sind von Interesse, um für die in Abschnitt 2.4 vorgestellten Perturbationen festzustellen, wie groß der Parameter k jeweils maximal gewählt werden darf.
Zudem werden die Algorithmen für überbestimmte Systeme auch für eine der in Kapitel 5 beschriebenen Attacken benötigt. In diesem Kapitel werden zwei Arten von Attacken auf HFE beschrieben, eine Attacke direkt auf das HFEP und eine auf das zugrunde liegende Problem MinRank, mit der versucht wird, den geheimen Schlüssel zu rekonstruieren.

Kapitel 3

Gröbnerbasen

In diesem Kapitel wird eine Möglichkeit untersucht, wie auf ziemlich allgemeine Art versucht werden kann, das bei HFE auftretende Gleichungssystem zu lösen.
Eine weit verbreitete Vorgehensweise, polynomiale Gleichungssysteme über beliebigen Körpern zu lösen, ist es, eine Gröbnerbasis des Ideals zu berechnen, das dem Gleichungssystem zugrunde liegt, und ähnlich der Gauss-Elimination die Lösungen abzulesen.
Der Begriff der Gröbnerbasis geht auf Bruno Buchberger zurück, der 1965 in seiner Dissertation [Bu65] einen Algorithmus zur Bestimmung der Basiselemente eines Polynomideals – den später nach ihm benannten Buchberger-Algorithmus — beschreibt und die dabei auftretenden Idealbasen nach seinem Doktorvater Wolfgang Gröbner benennt.

In diesem Kapitel werden zunächst in Abschnitt 3.1 die Grundlagen über Gröbnerbasen erläutert. Im anschließenden Abschnitt wird beschrieben, wie Gröbnerbasen mit Hilfe des Buchberger-Algorithmus bestimmt werden können. Zudem werden die grundlegenden Ergebnisse über die allgemeine Komplexität des Buchberger-Algorithmus zusammengefasst. Anschließend wird in Abschnitt 3.3 dargelegt, wie lexikographische Gröbnerbasen dazu verwendet werden können, Gleichungssysteme zu lösen. Da die Berechnung lexikographischer Gröbnerbasen allerdings mit dem Buchberger-Algorithmus vermutlich ineffizienter ist als beispielsweise die Berechnung graduierter Gröbnerbasen, wird zudem der FGLM-Algorithmus vorgestellt, mit dem im Spezialfall nulldimensionaler Ideale verschiedene Gröbnerbasen ineinander überführt werden können.
Im letzten Abschnitt 3.4 wird dann der Bezug zu HFE hergestellt. Da der Buchberger-Algorithmus im Allgemeinen eine sehr hohe (doppelt exponentielle) Komplexität hat, wird in Abschnitt 3.4.1 zunächst untersucht, inwiefern sich diese Komplexität durch Einschränkung auf den Spezialfall eines HFE-Systems senken lässt. Im abschließenden Abschnitt 3.4.2 werden dann die Ergebnisse von eigenen, praktischen Simulationen vorgestellt und analysiert, in denen untersucht wurde, wie effizient HFE-Systeme mit Gröbnerbasistechniken zu lösen sind.

3.1 Grundlagen

Die Gröbnerbasistheorie wird in dieser Arbeit ausschließlich für Polynomringe über Körpern beschrieben. Die Theorie könnte auch allgemeiner für Polynomringe über Ringen aufgestellt werden, aber diese Beschreibung wäre deutlich aufwendiger und ist für die Anwendung auf HFE unnötig.

Soweit nicht anders angegeben, werden in diesem Kapitel folgende Bezeichnungen verwendet:
K sei ein beliebiger Körper, $K[X] := K[x_1, \ldots, x_n]$ der Polynomring über K in n Veränderlichen und $\mathcal{T} := \left\{ x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \mid i_1, \ldots, i_n \in \mathbb{N}_0 \right\}$ sei der Monoid aller Terme in $K[X]$.
Für eine Menge von Polynomen $A \subseteq K[X]$ und einen beliebigen Erweiterungskörper L von K bezeichne $V_L(A) := \{\xi = (\xi_1, \ldots, \xi_n) \in L^n \mid f(\xi) = 0 \ \forall f \in A\}$ die Varietät von A über L. Ist $A = \{f_1, \ldots, f_m\}$ eine endliche Menge, so wird $V_L(f_1, \ldots, f_m) := V_L(A)$ auch als Lösungsgesamtheit des Polynomsystems $\{f_1, \ldots, f_m\}$ bezeichnet. Im Falle $L = K$ wird auch $V_K(A) := V(A)$ und $V(f_1, \ldots, f_m) := V_K(f_1, \ldots, f_m)$ verwendet.

Ein Ziel in diesem Kapitel ist, die Lösungsmengen von polynomialen Gleichungssystemen, also Varietäten, zu bestimmen. Die Idee dabei ist, zu einem gegebenen Polynomsystem, das davon erzeugte Ideal zu betrachten und dann die Varietät dieses Ideals zu bestimmen. Diese stimmt nämlich mit der Lösungsgesamtheit des Polynomsystems überein, wie der folgende Satz zeigt:

Satz 3.1.1
Seien $f_1, \ldots, f_m \in K[X]$. Dann ist

$$V(f_1, \ldots, f_m) \quad = \quad V((f_1, \ldots, f_m)).$$

Beweis:
$V(f_1, \ldots, f_m) \supseteq V((f_1, \ldots, f_m))$ ist wegen $f_1, \ldots, f_m \in (f_1, \ldots, f_m)$ klar.
Für $\xi \in V(f_1, \ldots, f_m)$ und $g = \sum h_i \cdot f_i \in (f_1, \ldots, f_m)$ ist auch $g(\xi) = \sum h_i(\xi) \cdot \underbrace{f_i(\xi)}_{=0} = 0$,
also $\xi \in V((f_1, \ldots, f_m))$. ■

Damit folgt sofort, dass zwei Polynomsysteme, die das gleiche Ideal erzeugen, die gleiche Lösungsgesamtheit haben:

Korollar 3.1.2
Seien $f_1, \ldots, f_m, g_1, \ldots, g_r \in K[X]$. Dann gilt

$$(f_1, \ldots, f_m) = (g_1, \ldots, g_r) \quad \Rightarrow \quad V(f_1, \ldots, f_m) = V(g_1, \ldots, g_r).$$

Nach diesem Korollar besteht eine Möglichkeit zur Lösung von polynomialen Gleichungssystemen darin, zu versuchen, eine Basis des vom Polynomsystem erzeugten Ideals zu bestimmen, aus der sich die Lösungen in vergleichsweise einfacher Weise ablesen lassen. Diese Eigenschaft haben beispielsweise lexikographische Gröbnerbasen (siehe auch Abschnitt 3.3).
Im univariaten Fall, d.h. über dem Ring $K[x_1]$ ist dieses Verfahren wohlbekannt:
Da $K[x_1]$ ein Hauptidealring ist, kann zu jedem Ideal $(f_1, \ldots, f_m)$ eine Idealbasis aus genau einem Element $g \in K[x_1]$, nämlich $g = ggT(f_1, \ldots, f_m)$, gefunden werden. Die gesuchte einfache Idealbasis kann also leicht mit Hilfe des Euklidischen Algorithmus bestimmt werden.
Tatsächlich kann der Buchberger-Algorithmus auch als Verallgemeinerung des Euklidischen Algorithmus auf den multivariaten Fall aufgefasst werden. Dazu werden im nächsten Abschnitt zunächst geeignete Ordnungen auf $\mathcal{T}$ definiert.

3.1.1 Termordnungen

Ein wesentlicher Aspekt, der dem Euklidischen Algorithmus zugrunde liegt, ist die durch die Gradfunktion gegebene, natürliche Ordnung auf den Termen in $K[x_1]$. Denn beim Euklidischen Algorithmus wird systematisch in jedem Schritt der Term höchsten Grades, der sogenannte Leitterm, durch geschicktes Verknüpfen der gegebenen Polynome beseitigt und so der Grad der Polynome in jedem Schritt verkleinert. Dadurch ist gesichert, dass der Algorithmus terminiert, denn die zugrunde liegende Ordnung, nämlich die natürliche Ordnung auf $\mathbb{N}_0$, ist eine Wohlordnung, d.h. jede strikt absteigende Kette von Elementen ist endlich.

Da der Buchberger-Algorithmus als Verallgemeinerung des Euklidischen Algorithmus betrachtet werden kann, ist es naheliegend, zunächst auch für die Menge $\mathcal{T}$ der multivariaten Terme in $K[X]$ Ordnungen zu definieren, die ähnliche Eigenschaften haben:

Definition 3.1.3
Sei $<$ eine totale Ordnung auf $\mathcal{T}$. Dann heißt $<$ eine ***Termordnung*** *oder* ***Reduktionsordnung****, falls gilt:*

- *Für alle $t \in \mathcal{T}$ ist $1 \leq t$.*
- *Für alle $t, t_1, t_2 \in \mathcal{T}$ mit $t_1 \leq t_2$ folgt $t \cdot t_1 \leq t \cdot t_2$.*

Bemerkung 3.1.4
Für jede Termordnung $<$ gilt nach obiger Definition

$$t \mid t' \quad \Rightarrow \quad t \leq t'.$$

Termordnungen bilden also eine Verfeinerung der durch die Teilbarkeitsrelation gegebenen partiellen Ordnung auf $\mathcal{T}$.

Zusätzlich haben die so definierten Termordnungen auch die wichtige Wohlordnungseigenschaft, wie der folgende Satz zeigt:

Satz 3.1.5
Sei $<$ eine Termordnung auf $\mathcal{T}$ und $t_1 \geq t_2 \geq t_3 \geq \ldots$ eine bezüglich dieser Ordnung absteigende Kette von Elementen aus $\mathcal{T}$. Dann existiert ein $s \in \mathbb{N}$ mit $t_j = t_s$ für alle $j \geq s$.
Beweis:
Die Ideale $\mathfrak{a}_i := (t_1, \ldots, t_i)$ bilden eine aufsteigende Kette. Da $K[X]$ noethersch ist, existiert ein $s \in \mathbb{N}$ mit $\mathfrak{a}_j = \mathfrak{a}_s$ für alle $j \geq s$. Damit ist $t_j \in \mathfrak{a}_s$, also

$$t_j = \sum_{i=1}^{s} \left(\sum_k c_{ik} q_{ik} \right) \cdot t_i$$

für Koeffizienten $c_{ik} \in K$ und Terme $q_{ik} \in \mathcal{T}$.
Angenommen, es wäre $t_i > t_j$ für $i = 1, \ldots, s$. Dann folgte $q_{ik} t_i > q_{ik} t_j \geq t_j$ für alle q_{ik} und damit $t_j \neq \sum_{i=1}^{s} \sum_k c_{ik} \cdot q_{ik} t_i$ im Widerspruch zur obigen Feststellung.
Damit ist also $t_i \leq t_j$ und wegen $t_1 \geq t_2 \geq t_3 \geq \ldots$ folgt die Behauptung. ■

Korollar 3.1.6
Zu einer Termordnung $<$ besitzt jede Teilmenge von $\mathcal{T}$ ein kleinstes Element bezüglich dieser Ordnung.

Im Folgenden werden nun zunächst die gebräuchlichsten Termordnungen vorgestellt:

Am natürlichsten ist vermutlich die lexikographische Termordnung $<_{lex}$. Sie wird als lexikographisch bezeichnet, da sie an die auf Wörtern übliche Ordnung angelehnt ist, in der zunächst nach dem ersten Buchstaben geordnet wird, bei Gleichheit dann nach dem zweiten usw. An die Stelle der Positionen im Wort treten hier die verschiedenen Variablen und an die Stelle der verschiedenen Buchstaben treten die jeweiligen Grade. Die Terme werden beispielsweise zunächst nach ihrem Grad in x_1 geordnet. Falls sie in x_1 den gleichen Grad haben, werden sie nach dem Grad in x_2 sortiert usw. Mit $x^\alpha := x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$ für $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ kann die lexikographische Ordnung definiert werden durch

Definition 3.1.7
*Die **lexikographische Termordnung** $<_{lex}$ wird definiert durch*

$$x^\alpha <_{lex} x^\beta \quad :\Leftrightarrow \quad \exists k \in \{1, \ldots n\} : \ \alpha_i = \beta_i \text{ für } i = 1, \ldots, k-1 \quad \text{und } \alpha_k < \beta_k.$$

Genau genommen ist dies die Definition der lexikographischen Termordnung mit $x_1 > x_2 > \ldots > x_n$. Lexikographische Termordnungen können natürlich auch für alle Permutationen der Reihenfolgen in den Variablen definiert werden. Insgesamt gibt es somit $n!$ verschiedene lexikographische Termordnungen.

Eine weitere Klasse von Termordnungen bilden die sogenannten graduierten Termordnungen. Bei diesen graduierten Ordnungen werden die Terme zunächst nach ihrem Gesamtgrad sortiert, d.h. graduierte Ordnungen sind Verfeinerungen der partiellen Ordnung, die natürlicherweise durch den Gesamtgrad der Terme gegeben ist. Ist der Gesamtgrad zweier Terme gleich, wird der Vergleich durch ein weiteres Kriterium, beispielsweise durch eine andere Termordnung, entschieden.
Graduierte Ordnungen haben den Vorteil, dass durch einen Term nach oben beschränkte Teilmengen, also Teilmengen der Form $\{t \in \mathcal{T} \mid t \leq t'\}$ für ein $t' \in \mathcal{T}$, immer endlich sind. Dies gilt beispielsweise für lexikographische Ordnungen im Allgemeinen nicht.
Als zweites Entscheidungskriterium nach dem Gesamtgrad kann z.B. eine lexikographische Ordnung gewählt werden. Dies ergibt dann eine graduiert-lexikographische Ordnung:

Definition 3.1.8
Die ***graduiert-lexikographische Termordnung*** $<_{glex}$ *wird definiert durch*

$$x^\alpha <_{glex} x^\beta \quad :\Leftrightarrow \quad \begin{array}{l} |\alpha| := \sum_{i=1}^n \alpha_i \ < \ \sum_{i=1}^n \beta_i =: |\beta| \qquad \textit{oder} \\ |\alpha| = |\beta| \textit{ und } \exists k \in \{1,\ldots,n\} : \ \alpha_i = \beta_i \textit{ für } i = 1,\ldots,k-1 \textit{ und } \alpha_k < \beta_k. \end{array}$$

Eine gebräuchliche, weil in der Praxis sehr effiziente, Termordnung ist die sogenannte graduierte umgekehrt-lexikographische Ordnung $<_{grevlex}$ (für „**g**raded **rev**erse **lex**ikographical“). Hier wird als zweites Entscheidungskriterium die lexikographische Ordnung gleich in zweifacher Weise umgekehrt: Grob gesprochen wird zunächst nach den hinteren Variablen, d.h. nach $x_n, x_{n-1}, \ldots$ sortiert, und kleinere Grade in diesen Variablen ergeben größere Terme. In der exakten Formulierung bedeutet dies:

Definition 3.1.9
Die ***graduierte umgekehrt-lexikographische Termordnung*** $<_{grevlex}$ *wird definiert durch*

$$x^\alpha <_{grevlex} x^\beta \quad :\Leftrightarrow \quad \begin{array}{l} |\alpha| \ < \ |\beta| \qquad \textit{oder} \\ |\alpha| = |\beta| \textit{ und } \exists k \in \{1,\ldots,n\} : \ \alpha_i = \beta_i \textit{ für } i = k+1,\ldots,n \textit{ und } \alpha_k > \beta_k. \end{array}$$

Bemerkung 3.1.10
Auf den ersten Blick scheint diese doppelte Umkehrung wieder die oben beschriebene Ordnung $<_{glex}$ *zu ergeben. Es ist aber leicht, sich davon zu überzeugen, dass sich diese*

Ordnungen für $n \geq 3$ unterscheiden, denn mit $\alpha = (1,4,3)$ und $\beta = (2,2,4)$ gilt beispielsweise

$$\begin{aligned} & x_1 x_2^4 x_3^3 && <_{glex} && x_1^2 x_2^2 x_3^4, && \text{da } \alpha_1 < \beta_1 \text{ ist,} \\ \text{und} \quad & x_1 x_2^4 x_3^3 && >_{grevlex} && x_1^2 x_2^2 x_3^4, && \text{da } \alpha_3 < \beta_3 \text{ gilt.} \end{aligned}$$

Mit Termordnungen können unter anderem die Terme innerhalb eines Polynoms aus $K[X]$ sortiert werden. Insbesondere ist dadurch ein Term, nämlich der größte bezüglich der vorgegebenen Ordnung, ausgezeichnet. Da dieser Term und der dazugehörige Koeffizient in den folgenden Abschnitten große Bedeutung haben, bekommen sie eine eigene Bezeichnung:

Definition 3.1.11
Sei $<$ eine Termordnung auf $\mathcal{T}$ und $f = \sum c_\alpha \cdot x^\alpha \in K[X]$.

- $lt(f) := \max_< \{x^\alpha \mid c_\alpha \neq 0\}$ *ist der* ***Leitterm*** *von f.*
- $lc(f) := c_\alpha$ *mit α so, dass $lt(f) = x^\alpha$ gilt, heißt der* ***Leitkoeffizient*** *von f.*
- $lm(f) := lc(f) \cdot lt(f)$ *wird das* ***Leitmonom*** *von f genannt.*

Für das Nullpolynom wird $lt(0) := -\infty$ und $lc(0) := 0$ vereinbart.

Sofern nichts anderes angegeben ist, sei in den folgenden Abschnitten immer eine beliebige Termordnung $<$ auf $\mathcal{T}$ fest vorgegeben.

3.1.2 Reduktionen

In diesem Abschnitt wird nun der Hauptbestandteil des Euklidischen Algorithmus, die Polynomdivision, auf den multivariaten Fall verallgemeinert. Dies ergibt die sogenannte Reduktion, die manchmal auch als multivariate Polynomdivision bezeichnet wird.

Für $f, g \in K[x_1]$ ist bekannt, dass sich eindeutig bestimmte $q, r \in K[x_1]$ finden lassen mit

$$f = q \cdot g + r \qquad \text{und} \quad (\ r = 0 \ \text{ oder } \ \deg(r) < \deg(g)\ ).$$

Wie schon zu Anfang des letzten Abschnittes festgestellt, liegt diesem Verfahren die auf den univariaten Termen natürliche, durch ihren Grad gegebene Ordnung zugrunde. Diese ist insbesondere auch eine spezielle Termordnung (die einzige auf $K[x_1]$) und die Bedingung $\deg(r) < \deg(g)$ ist in den Begriffen des letzten Kapitels äquivalent zu $lt(g) \nmid lt(r)$.

In dieser Form lässt sich das univariate Problem der Polynomdivision fast direkt auf den multivariaten Fall übertragen:
Seien $f, f_1, \ldots, f_m \in K[X]$ gegeben. Gesucht ist eine Darstellung der Form

$$f = \sum_{i=1}^{m} q_i \cdot f_i + r \qquad \text{mit } lt(f_i) \nmid lt(r) \text{ für } i = 1, \ldots, m.$$

Für die univariate Polynomdivision ist bekannt, dass q und r für gegebene f und g eindeutig bestimmt sind. Dies ist bei dieser multivariaten Problemstellung leider nicht der Fall:
Wenn beispielsweise $lt(f_i) < lt(r)$ ist, dann ist eine weitere, gültige Darstellung gegeben durch $q_i' := q_i - 1$ und $r' := r + f_i$, da $lt(r') = lt(r)$ gilt.
Zudem kann es bei der Darstellung von f durch mehrere Polynome $f_1, \ldots, f_m$ durch gegenseitige Auslöschung von Leitmonomen der verschiedenen $q_i f_i$ passieren, dass einzelne $q_i f_i$ einen größeren Leitterm haben als das darzustellende f. Solche Darstellungen sind aber unnötig kompliziert, da der unten folgende Algorithmus zeigt, dass es immer auch eine Darstellung mit $lt(f) \geq lt(q_i \cdot f_i)$ gibt.

Um zumindest diese Freiheiten bei der Wahl der q_i und r zu vermeiden, wird die obige Problemstellung für die multivariate Polynomdivision noch um zwei Bedingungen ergänzt:

Seien $f \in K[X]$ und $B = \{f_1, \ldots, f_m\} \subset K[X]$ gegeben. Gesucht ist eine Darstellung der Form

$$f = \sum_{i=1}^{m} q_i \cdot f_i + r, \tag{3.1}$$

so dass für $i = 1, \ldots, m$ gilt

$$lt(f) \geq lt(q_i \cdot f_i) \quad \text{und} \quad lt(f_i) \text{ teilt keinen der Terme in } r. \tag{3.2}$$

Diese Formulierung ist ebenfalls eine Verallgemeinerung der bekannten univariaten Polynomdivision, da die zusätzlichen Bedingungen dort trivialerweise erfüllt sind.
Wie eine solche Darstellung gefunden werden kann, zeigt der folgende Algorithmus:

Algorithmus 3.1.12 (Reduktion / Multivariate Polynomdivision)
Eingabe:
$f \in K[X],\ B = \{f_1, \ldots, f_m\} \subseteq K[X]$
Ausgabe:
$r \in K[X]$, so dass $q_1, \ldots, q_m \in K[X]$ existieren, mit denen (3.1) und (3.2) erfüllt sind

Reduktion(f, B):
Sei $r := 0$, $h := f$
Wiederhole
Falls $lt(f_i) \nmid lt(h)$ für $i = 1, \ldots, m$
dann $r := r + lm(h)$
$h := h - lm(h)$
sonst wähle ein $i \in \{1, \ldots, m\}$ mit $lt(f_i) \mid lt(h)$
$h := h - \frac{lm(h)}{lm(f_i)} \cdot f_i$ $(*)$
bis $h = 0$
return r

Zur Korrektheit des Algorithmus: Der Algorithmus terminiert, da in jedem Schleifendurchlauf das Polynom h entweder durch $h - lm(h)$ oder durch $h - \frac{lm(h)}{lm(f_i)} \cdot f_i$ ersetzt wird. In beiden Fällen wird h also — falls $h \neq 0$ ist — durch ein Polynom ersetzt, dessen Leitterm bezüglich der gegebenen Termordnung echt kleiner ist als der Leitterm von h. Da Termordnungen aber nach Satz 3.1.5 immer Wohlordnungen sind, bricht diese Kette nach endlich vielen Schritten ab, d.h. h wird 0.
Zudem sind auch die Bedingungen (3.1) und (3.2) für das zurückgegebene r erfüllt: Denn wird in Zeile $(*)$ zusätzlich noch $q_i := q_i + \frac{lm(h)}{lm(f_i)}$ gesetzt (für anfangs mit 0 initialisierte q_i), so gilt nach jedem Schleifendurchlauf für die aktuelle Belegung der Variablen

$$h + \sum_{i=1}^{m} q_i \cdot f_i + r = f.$$

Insbesondere ist nach dem letzten Durchlauf also Bedingung (3.1) erfüllt. Dabei kommen zu r nur Terme hinzu, die nicht durch Leitterme der f_i geteilt werden und die obige Beschreibung der q_i zeigt, dass auch immer $lt(q_i \cdot f_i) \leq lt(f)$ gilt, d.h. auch (3.2) ist erfüllt.

Polynome, die durch den Algorithmus Reduktion nicht verändert werden, bekommen eine eigene Bezeichnung:

Definition 3.1.13
*$f \in K[X]$ heißt **reduziert** bezüglich $f_1, \ldots, f_m \in K[X]$, falls*

$$Reduktion(f, \{f_1, \ldots, f_m\}) = f$$

gilt, d.h. falls die in f vorkommenden Terme von keinem der Leitterme der f_i geteilt werden.

3.1.3 Gröbnerbasen

Der Algorithmus $Reduktion(f, B)$ liefert immer ein r zurück, das wegen Bedingung (3.2) bezüglich B reduziert ist und wegen Bedingung (3.1) zur gleichen Nebenklasse in $K[X]/\mathfrak{a}$ mit $\mathfrak{a} := (B)$ wie das Eingabepolynom f gehört. Dies gibt Anlass zu der Hoffnung, dass mit Hilfe dieses Algorithmus eindeutige Repräsentanten der Nebenklassen in $K[X]/\mathfrak{a}$ gefunden werden können. Leider hängt der Algorithmus aber nicht direkt vom Ideal $\mathfrak{a}$ ab, das den Faktorring $K[X]/\mathfrak{a}$ bestimmt, sondern vielmehr von der Wahl der Basis B von $\mathfrak{a}$.
Eine sinnvolle Forderung für den gesuchten Repräsentanten einer Nebenklasse ist somit, dass er nicht nur bezüglich einer Basis B von $\mathfrak{a}$ reduziert ist, sondern bezüglich jeder beliebigen Basis von $\mathfrak{a}$. Er sollte also nur Terme enthalten, die nicht durch Leitterme von irgendwelchen Polynomen in $\mathfrak{a}$ teilbar sind. Dies führt zu folgenden Begriffsbildungen:

Definition 3.1.14
Sei $\mathfrak{a} \subseteq K[X]$ *ein Ideal und* $f \in K[X]$.

- *Die Menge* $\mathcal{N} := \mathcal{N}(\mathfrak{a}) := \mathcal{T} \setminus (lt(\mathfrak{a}))$ *mit* $lt(\mathfrak{a}) := \{lt(f) \mid f \in \mathfrak{a}\}$ *heißt die* ***Normalmenge*** *von* $\mathfrak{a}$.
- *Ein Repräsentant* $r \in f + \mathfrak{a}$, *dessen Terme alle in* $\mathcal{N}$ *liegen, d.h.* $r \in \langle \mathcal{N} \rangle_K \cap f + \mathfrak{a}$, *wird eine* ***Normalform*** *von* f *bezüglich* $\mathfrak{a}$ *genannt.*

Im Folgenden ist es das Ziel, zu zeigen, dass mit Hilfe des Reduktionsalgorithmus Normalformen berechnet werden können und dass diese eindeutig bestimmt sind.
Das Ergebnis r eines Aufrufes von $Reduktion(f, B)$ ist genau dann *keine* Normalform, wenn es einen Term aus dem sogenannten Leittermideal $(lt(\mathfrak{a}))$ enthält. Um diesen Fall zu vermeiden, ist es also sinnvoll, die Basis $B = \{f_1, \ldots, f_m\}$ von $\mathfrak{a}$ so zu wählen, dass alle Terme aus $(lt(\mathfrak{a}))$ durch die Leitterme der f_i teilbar sind. Dies führt zum Begriff der Gröbnerbasis:

Definition 3.1.15
Sei $\mathfrak{a} \subseteq K[X]$ *ein Ideal und* $G := \{g_1, \ldots, g_m\} \subset \mathfrak{a}$.
G *heißt eine* ***Gröbnerbasis*** *von* $\mathfrak{a}$ *(bezüglich der Termordnung* $<$*), falls das Leittermideal von* $\mathfrak{a}$ *von den Leittermen der* g_i *erzeugt wird, d.h.*

$$(lt(g_1), \ldots, lt(g_m)) \quad = \quad (lt(\mathfrak{a})).$$

Bemerkung 3.1.16

- *Nach dem Hilbertschen Basissatz wird* $(lt(\mathfrak{a}))$ *von einer endlichen Teilmenge von* $lt(\mathfrak{a})$ *erzeugt. Also besitzt jedes Ideal eine Gröbnerbasis.*
- *Der Begriff Gröbner*basis *ist gerechtfertigt, da* G *wirklich eine Basis von* $\mathfrak{a}$ *ist wie in Satz 3.1.18 gezeigt wird.*

Folgendes Lemma zeigt, dass Gröbnerbasen wirklich das Gewünschte leisten:

Lemma 3.1.17
Sei $G = \{g_1, \ldots, g_m\}$ *eine Gröbnerbasis von* $\mathfrak{a} \subseteq K[X]$. *Dann gilt für jeden Term* $t \in \mathcal{T}$:

$$t \in (lt(\mathfrak{a})) \qquad \Leftrightarrow \qquad \exists\, i \in \{1, \ldots, m\} : \quad lt(g_i) \mid t.$$

Beweis:
Die Richtung „$\Leftarrow$“ ist nach Definition der Gröbnerbasis klar.
Sei also $t \in (lt(\mathfrak{a}))$ ein Term. Dann existieren $q_1, \ldots, q_m \in K[X]$ mit $t = \sum q_i \cdot lt(g_i)$, da G eine Gröbnerbasis von $\mathfrak{a}$ ist. Da t ein Term ist, ist aber auch

$$t = lt(t) = lt\left(\sum q_i \cdot lt(g_i)\right) = t' \cdot lt(g_i)$$

für einen Term t', der in q_i für ein $i \in \{1, \ldots, m\}$ vorkommt. Also wird t durch $lt(g_i)$ geteilt. ■

Ist eine Gröbnerbasis des Ideals $\mathfrak{a}$ bekannt, kann also die Normalmenge leicht charakterisiert werden als die Menge derjenigen Terme, die nicht durch die Leitterme der Gröbnerbasiselemente teilbar sind. Ferner können dann Normalformen mit Hilfe des Reduktionsalgorithmus berechnet werden, die zudem noch eindeutig bestimmt sind, wie der folgende Satz zeigt:

Satz 3.1.18
Sei $G = \{g_1, \ldots, g_m\}$ eine Gröbnerbasis eines Ideals $\mathfrak{a} \subseteq K[X]$ und $f \in K[X]$. Dann gilt

i) $\mathcal{N} = \{t \in \mathcal{T} \mid lt(g_i) \nmid t \text{ für } i = 1, \ldots, m\}$,

ii) G ist eine Basis von $\mathfrak{a}$, d.h. $(G) = \mathfrak{a}$,

iii) Es gibt genau eine Normalform von f bezüglich $\mathfrak{a}$ und diese kann durch $Reduktion(f, G)$ berechnet werden.

Beweis:
i) folgt direkt aus der Definition von $\mathcal{N}$ und Lemma 3.1.17.
zu *ii*): $(G) \subseteq \mathfrak{a}$ folgt aus der Definition der Gröbnerbasis.
Sei nun $g \in \mathfrak{a}$ und $r := Reduktion(g, G)$. Dann ist $r = g - \sum q_i g_i \in \mathfrak{a}$, d.h. falls $r \neq 0$ ist, existiert nach Lemma 3.1.17 ein $i \in \{1, \ldots, m\}$ mit $lt(g_i) \mid lt(r)$. Andererseits ist r aber reduziert bezüglich G, d.h. es muss $r = 0$ und damit $g \in (G)$ sein.
zu *iii*): Sei $r := Reduktion(f, G)$. Wegen Bedingung (3.1) ist dann $r \in f + (G) \overset{ii)}{=} f + \mathfrak{a}$ und nach *i*) folgt aus Bedingung (3.2) direkt, dass auch $r \in \langle \mathcal{N} \rangle_K$ gilt. Also ist r eine Normalform von f bezüglich $\mathfrak{a}$.
Sei nun $r' \in K[X]$ eine weitere Normalform von f bezüglich $\mathfrak{a}$. Dann ist wegen $r' \in f + \mathfrak{a}$ insbesondere $r - r' \in \mathfrak{a}$, also $lt(r - r') \in (lt(\mathfrak{a}))$. Zudem ist aber auch $r - r' \in \langle \mathcal{N} \rangle_K$, also $lt(r - r') \in (lt(\mathfrak{a})) \cap \langle \mathcal{N} \rangle_K = \{0\}$. Also folgt $r = r'$ und damit die Behauptung. ■

Definition 3.1.19
Die nach Satz 3.1.18 eindeutig bestimmte Normalform eines Polynoms $f \in K[X]$ bezüglich eines Ideals $\mathfrak{a} \subseteq K[X]$ wird mit $NF(f, \mathfrak{a})$ bezeichnet.

Gröbnerbasen liefern also ein gutes Kriterium, um zu überprüfen, ob ein Polynom in einem gegebenen Ideal liegt:

Korollar 3.1.20
Für jede Gröbnerbasis G eines Ideals $\mathfrak{a} \subseteq K[X]$ und $f \in K[X]$ gilt

$$f \in \mathfrak{a} \qquad \Leftrightarrow \qquad Reduktion(f, G) = 0.$$

Beweis:
Die Äquivalenz folgt direkt aus $Reduktion\,(f,G) = NF\,(f,\mathfrak{a}) \in f + \mathfrak{a}$. ■

Es lassen sich zwei besondere Arten von Gröbnerbasen charakterisieren:

Definition 3.1.21
Sei $G = \{g_1, \ldots, g_m\}$ eine Gröbnerbasis des Ideals $\mathfrak{a} \subseteq K[X]$.

*i) G heißt **minimal**, falls $(G \setminus \{g_i\}) \subsetneq (G)$ für alle $i = 1, \ldots, m$ gilt.*

*ii) G heißt **reduziert**, wenn für $i = 1, \ldots, m$ gilt:*

$$lc\,(g_i) = 1 \text{ und } g_i \text{ ist reduziert bezüglich } G \setminus \{g_i\}\,.$$

Bemerkung 3.1.22
Zu jedem Ideal und jeder Termordnung existiert genau eine reduzierte Gröbnerbasis (siehe beispielsweise [CLO92]).

3.2 Bestimmung von Gröbnerbasen

Damit bleibt noch die Frage, wie Gröbnerbasen bestimmt werden können. Eine Möglichkeit dazu bietet der Buchberger-Algorithmus, der in diesem Abschnitt beschrieben wird.

3.2.1 Buchberger-Algorithmus

Eine gegebene Basis $B = \{f_1, \ldots, f_t\}$ eines Ideals ist genau dann *keine* Gröbnerbasis, wenn es eine Kombination $h = \sum q_i \cdot f_i$ von Elementen aus B gibt, so dass $lt\,(h)$ von keinem der Leitterme der f_i geteilt wird. Buchbergers entscheidende Idee war, möglichst einfache solche Kombinationen, für die dies auftreten kann, die sogenannten S-Polynome, zu betrachten:

Definition 3.2.1
*Seien $f, g \in K[X] \setminus \{0\}$. Dann ist das **S-Polynom** von f und g definiert als*

$$S\,(f,g) \quad := \quad \frac{kgV\,(lt\,(f)\,, lt\,(g))}{lm\,(f)} \cdot f \quad - \quad \frac{kgV\,(lt\,(f)\,, lt\,(g))}{lm\,(g)} \cdot g.$$

Das S-Polynom zu f und g ist also die einfachste Kombination von f und g, für die sich nicht a priori sagen lässt, ob $lt\,(S\,(f,g))$ durch einen der Leitterme von f und g teilbar ist, da sich die Leitmonome der Summanden gegenseitig auslöschen und darum der Leitterm echt kleiner ist als die Leitterme der beiden Summanden.
Tatsächlich sagt Buchbergers Kriterium nun aus, dass es ausreicht, nur die paarweisen S-Polynome einer Basis eines Ideals zu betrachten, um zu erkennen, ob diese Basis eine Gröbnerbasis ist:

Theorem 3.2.2 (Buchbergers Kriterium)
Sei $B = \{f_1, \ldots, f_m\} \subset K[X]$. *Dann sind äquivalent:*

i) B *ist Gröbnerbasis von* $(f_1, \ldots, f_m)$,

ii) *Reduktion* $(S(f_i, f_j), B) = 0$ *für alle* $1 \leq i < j \leq m$.

Beweis:
„$i) \Rightarrow ii)$“ folgt aus Korollar 3.1.20, da natürlich $S(f_i, f_j) \in (B)$ ist.
Ausführliche Beweise der Rückrichtung stehen beispielsweise in [CLO92] und [GG99]. ■

Dieses Kriterium kann nun in einfacher Art und Weise für einen Algorithmus zur Bestimmung von Gröbnerbasen verwendet werden:

Algorithmus 3.2.3 (Buchberger-Algorithmus)
Eingabe:
$B = \{f_1, \ldots, f_m\} \subseteq K[X]$
Ausgabe:
Gröbnerbasis G des Ideals (B)

Buchberger(B):
Sei $P := \{(i,j) \mid 1 \leq i < j \leq m\}$, $G := B$, $m' := m$
Wiederhole
Wähle ein Paar $(i,j) \in P$ und entferne es aus P
Sei $h := Reduktion(S(f_i, f_j), G)$
Wenn $h \neq 0$
dann
Erhöhe m' um 1
Füge $f_{m'} := h$ zu G hinzu
Ergänze P um die Paare (i, m'), $i = 1, \ldots, m' - 1$
bis $P = \emptyset$
return G

Zur Korrektheit des Algorithmus:
Betrachte das Ideal $(lt(G)) := (\, lt(g) \mid g \in G \,)$, das von den Leittermen der in G enthaltenen Polynome erzeugt wird, im Verlauf des Algorithmus. Jedesmal, wenn zu G ein Polynom h hinzugefügt wird, wird $(lt(G))$ echt größer, da h das Ergebnis einer Reduktion bezüglich G ist und somit sein Leitterm noch nicht in $(lt(G))$ enthalten sein kann.
Da $K[X]$ noethersch ist, wird das Ideal $(lt(G))$ nach endlich vielen Schritten des Algorithmus stationär, d.h. zu G werden nach endlich vielen Schritten keine Polynome mehr hinzugefügt. Ab diesem Zeitpunkt wird die endliche Menge P aber in jedem Schleifendurchlauf um ein Element kleiner und somit terminiert der Algorithmus.

Zudem wird im Verlauf des Algorithmus jedes Paar (f_i, f_j) von Polynomen aus dem am Ende zurückgelieferten G betrachtet. Sei G' die Menge, die G in dem Schritt entspricht, zu dem das Paar (f_i, f_j) betrachtet wurde. Dann ist $G' \subseteq G$ und für $h := \mathit{Reduktion}\left(S\left(f_i, f_j\right), G'\right)$ ist entweder $h = 0$ oder h wird im nächsten Schritt zu G' hinzugefügt, d.h. $G \supseteq G' \cup \{h\}$. In jedem Fall ist $\mathit{Reduktion}\left(S\left(f_i, f_j\right), G\right) = 0$, G also nach Buchbergers Kriterium eine Gröbnerbasis von (G).
Letztendlich ist zu jedem Zeitpunkt aber auch $(G) = (B)$, da G im Vergleich zu B nur zusätzliche Polynome enthält, die ohnehin schon im Ideal (B) enthalten sind. Der Algorithmus ist also korrekt.

Der Algorithmus, so wie er hier der Übersicht halber notiert ist, bietet natürlich noch viel Raum für Verbesserungen.
Beispielsweise ist im Algorithmus nicht näher beschrieben in welcher Weise das Paar $(i, j) \in P$ ausgewählt werden soll. Dafür gibt es mittlerweile eine Reihe heuristischer Verfahren, die — auch abhängig von der Art der Eingaben — unterschiedlich gute Ergebnisse liefern. Buchberger selber schlägt in [Bu85] beispielsweise vor, (i, j) immer so zu wählen, dass $kgV\left(lt\left(f_i\right), lt\left(f_j\right)\right)$ möglichst klein ist. Dies hätte die Auswirkung, dass die zugehörigen S-Polynome dazu tendierten, nicht auf 0 reduzierbar zu sein. So könnten dann schnell viele Elemente der Gröbnerbasis gefunden werden und die weiteren Reduktionen würden beschleunigt.

Ein Großteil der Rechenzeit wird für die Reduktionen benötigt. Ein weiterer Ansatzpunkt für eine Beschleunigung des Algorithmus wäre also ein Kriterium mit dem schnell erkannt werden könnte, welche S-Polynome ohnehin auf 0 reduziert werden können. So wird in [CLO92] beispielsweise gezeigt, dass viele Paare (i, j) gar nicht betrachtet werden brauchen: Zum einen sind dies die Paare (i, j), für die die Leitterme der Polynome f_i und f_j teilerfremd sind. Zum anderen brauchen auch Paare (i, j) nicht betrachtet zu werden, für die es ein k gibt, so dass $kgV\left(lt\left(f_i\right), lt\left(f_j\right)\right)$ von $lt\left(f_k\right)$ geteilt wird und sowohl $S\left(f_i, f_k\right)$ als auch $S\left(f_j, f_k\right)$ schon betrachtet worden sind.

3.2.2 Komplexität

Schon am oben beschriebenen Beweis der Korrektheit des Buchberger-Algorithmus, lässt sich erahnen, dass dessen Komplexität nicht einfach abzuschätzen ist. Denn schon die Endlichkeit des Algorithmus beruht darauf, dass der zugrunde liegende Polynomring noethersch ist, die entstehende Kette von Idealen also *irgendwann* stationär wird. In diesem Abschnitt soll nun ein kurzer Überblick darüber gegeben werden, was über die Komplexität der Berechnung von Gröbnerbasen bekannt ist.

Es ist nicht überraschend, dass sich leicht zeigen lässt, dass die Berechnung von Gröbnerbasen $\mathcal{NP}$-hart ist. In [Cox98] wird beispielsweise gezeigt, wie das bekannter-

maßen $\mathcal{NP}$-vollständige 3-Färbbarkeitsproblem auf die Berechnung von Gröbnerbasen reduziert werden kann.
Zudem ist aus Korollar 3.1.20 klar, dass sich auch das sogenannte „ideal membership problem“ IM, also die Frage danach, ob ein Polynom $f \in K[X]$ in einem gegebenen Ideal $\mathfrak{a} \subseteq K[X]$ liegt, auf die Berechnung von Gröbnerbasen reduzieren lässt. In [MM82] haben Ernst W. Mayr und A. Meyer aber gezeigt, dass IM über dem Körper $K = \mathbb{Q}$ $\mathcal{EXPSPACE}$-vollständig ist. Zudem gilt dies nach [May97] sogar für beliebige, unendliche Körper K. Nach der Hierarchie in Kapitel 1.3.2 liegt die Berechnung von Gröbnerbasen über unendlichen Körpern damit also sogar oberhalb der Klasse der Probleme, die in einfach exponentieller Zeit gelöst werden können.

Im Wesentlichen scheinen zwei Dinge für diese große Komplexität verantwortlich zu sein: Zum einen ist es je nach Grundkörper so, dass sich die Darstellungen der in den Polynomen vorkommenden Koeffizienten in jedem Schritt vergrößern. Dies betrifft insbesondere unendliche Körper, da in diesen Körpern bei exakter Rechnung, die Genauigkeit der Darstellung immer angepasst werden muss. Da Ziel dieser Arbeit die Anwendung der Techniken auf endlichen Körpern ist, wird dieses Komplexitätsproblem bezüglich der Koeffizienten hier nicht weiter betrachtet.
Zum anderen können die Grade der an den Rechnungen beteiligten Polynome im Vergleich zum Grad der Eingabepolynome recht groß werden. Dies betrifft nicht nur die Polynome der letztendlich berechneten Gröbnerbasis, sondern es gibt unter Umständen auch als Zwischenergebnisse der Rechnungen Polynome von noch viel größerem Grad.

Im Folgenden werden nun die wichtigsten Ergebnisse in Bezug auf die Grade der Ausgabepolynome vorgestellt:
Das wohl bekannteste Beispiel stammt von Mayr und Meyer, die in [MM82] gezeigt haben, dass die reduzierte Gröbnerbasis eines Ideals, das von Polynomen in n Unbekannten vom Grad $\leq d$ erzeugt wird, ein Polynom vom Grad $d^{2^{O(n)}}$ enthalten kann. Von Dung T. Huynh stammt ein Beispiel in [Huy86], für das zu jeder graduierten Termordnung die minimale Gröbnerbasis $d^{2^{O(n)}}$ Elemente enthält.
Eine allgemeinere Möglichkeit ist es, die worst case Komplexität in Abhängigkeit von d und n durch eine Funktion $D(n,d)$ anzugeben. Dazu sei $D(n,d)$ der maximale Grad eines Polynoms, das in einer bezüglich einer beliebigen Termordnung reduzierten Gröbnerbasis eines beliebigen Ideals, das von Polynomen vom Grad $\leq d$ erzeugt wird, vorkommt. Für diese Funktion $D(n,d)$ sind zwei Schranken bekannt. In [Dub89] wird gezeigt, dass

$$D(n,d) \leq d^{2^n}$$

gilt, während H. Michael Möller und Teo Mora in [MM84] zeigen, dass ein $c > 0$ existiert, so dass

$$D(n,d) \geq c \cdot d^{2^n}$$

für genügend große d und n gilt.

Dies alles sind worst case Beispiele, die zeigen, dass die Berechnung einer Gröbnerbasis im schlechtesten Fall wohl doppelt exponentielle Komplexität hat. Etwas besser ist die Komplexität in einem in der Praxis ziemlich wichtigen Spezialfall, nämlich im Fall nulldimensionaler Ideale.
Ein Ideal $\mathfrak{a}$ wird als **nulldimensional** bezeichnet, wenn es die Krull-Dimension 0 hat. Dies ist insbesondere äquivalent dazu, dass $V_{\overline{K}}(\mathfrak{a})$ endlich ist, was auf viele in der Praxis relevante Fälle zutrifft.

Im nulldimensionalen Fall ist Folgendes über die Komplexität bekannt:
Nach [CGH88] ist die Berechnung einer Gröbnerbasis eines nulldimensionalen Ideals bezüglich der graduierten umgekehrt-lexikographischen Termordnung $<_{grevlex}$ (vgl. Definition 3.1.9) polynomiell in d^{n^2}. Daniel Lazard hat in [Laz83] gezeigt, dass sie sogar polynomiell in d^n ist, falls das Ideal auch projektiv nulldimensional ist. Die direkte Berechnung einer Gröbnerbasis bezüglich der lexikographischen Termordnung ist dagegen deutlich komplexer. Ebenfalls in [CGH88] wurde gezeigt, dass diese Komplexität $d^{O(n^3)}$ ist. In Abschnitt 3.3.1 wird allerdings ein Algorithmus vorgestellt, der diese Komplexität auf $d^{O(n^2)}$ senkt, indem zunächst eine graduierte Gröbnerbasis berechnet wird und dann mit Hilfe linearer Algebra daraus die gesuchte, lexikographische Gröbnerbasis konstruiert wird.

Alle bisher hier aufgezählten Komplexitäten haben gemeinsam, dass es worst case Abschätzungen sind. Das bedeutet, die genannten Sätze sagen meist nur aus, dass es ein spezielles Ideal gibt, für das die Berechnung einer Gröbnerbasis eine solch hohe Komplexität hat. Dies hat für die Praxis (vor allem in der Kryptographie) aber, wie schon in Abschnitt 1.3.2 erwähnt, nur eine geringe Bedeutung, denn es ist natürlich durchaus denkbar, dass für einige — oder vielleicht sogar für viele — spezielle Ideale die Berechnung einer Gröbnerbasis deutlich einfacher ist. Analysen dieses average cases sind aber ungleich schwieriger als die ebenfalls komplizierten worst case Analysen. Deshalb gibt es auch kaum handfeste Aussagen über das average case Verhalten.
In [CLO92] und [GG99] wird lediglich angedeutet, dass die Berechnung von Gröbnerbasen von Idealen, die von geometrischen Problemen stammen, im Allgemeinen etwas einfacher sein könnte, während die hier genannten worst case Beispiele eher von künstlich konstruierten, kombinatorischen Problemstellungen abstammen. Anlass zu dieser Hoffnung gibt die von David Bayer und Michael Stillman in [BS88] jedem Ideal zugeordnete Invariante m, die sogenannte Castelnuovo-Mumford regularity. Diese Invariante scheint für geometrische Probleme eher klein zu sein, während sie für die Beispiele von Mayr und Meyer exponentiell groß wird. Das entscheidende Ergebnis von Bayer und Stillman ist, dass nach einem generischen Koordinatenwechsel, die Grade der Polynome in einer

Gröbnerbasis bezüglich der graduierten umgekehrt-lexikographischen Termordnung durch m beschränkt sind. Dieses Ergebnis gibt zudem weiteren Anlass zu der Vermutung, dass die graduierte umgekehrt-lexikographische Termordnung in der Praxis das beste Laufzeitverhalten hat.

3.3 Lösen von Gleichungssystemen mit Gröbnerbasen

Schon in Abschnitt 3.1 wurde eine grundlegende Idee zur Lösung von polynomialen Gleichungssystemen beschrieben, nämlich zu versuchen, eine Basis des von den gegebenen Polynomen erzeugten Ideals zu finden, aus der sich die Lösungen leicht ablesen lassen. In zwei Fällen sind Verfahren zur Bestimmung solcher Basen wohlbekannt: im univariaten Fall $n = 1$, also $K[X] = K[x_1]$, und im linearen Fall, d.h. $grad(f_i) = 1$ für $i = 1, \ldots, m$:

Univariater Fall

Da $K[x_1]$ ein Hauptidealring ist, existiert zu jedem Ideal $(f_1, \ldots, f_m)$ eine einelementige Basis $\{f\}$ bestehend aus $f = ggT(f_1, \ldots, f_m)$. Diese Basis lässt sich leicht — wie schon in den vorherigen Abschnitten beschrieben — durch den Euklidischen Algorithmus bestimmen. Um aus dieser Basis dann die Lösungen des Systems abzulesen, kann zunächst die irreduzible Zerlegung $f = \prod p_i$ von f bestimmt werden. Jedes p_i bestimmt dann einen Erweiterungskörper von K, in dem ein Teil der Lösungen des Systems, eben die Nullstellen von p_i, liegen.
Insbesondere sind die Lösungen des Systems, die im Grundkörper liegen, durch die Linearfaktoren von f gegeben.

Linearer Fall

In diesem Fall ist ein System $\{f_1, \ldots, f_m\}$ mit $f_i = \sum_{j=1}^{n} a_{ij}x_j + b_i$ gegeben, das üblicherweise mit der Gaußelimination gelöst wird.
Das Prinzip der Gaußelimination ist es, sukzessive die Variablen zu eliminieren, d.h. die Polynome in der Basis so zu modifizieren, dass letztendlich eine sogenannte Dreiecksbasis $\{g_1, \ldots, g_{m'}\}$ entsteht. Das heißt, dass (nach Umnumerierung, so dass etwaige freie Parameter gerade durch $x_{m'+1}, \ldots, x_n$ gegeben sind) die Variable x_i nur noch in den Polynomen $g_1, \ldots, g_i$ vorkommt. Aus einer solchen Dreiecksbasis können die Lösungen dann durch Rückwärtseinsetzen, d.h. durch Lösen von Gleichungen in (abgesehen von den freien Parametern) einer Variablen, bestimmt werden.

Der Euklidische Algorithmus und das Vorgehen im univariaten Fall wurde in den vorigen Abschnitten schon zur Motivierung für die Bestimmung von Gröbnerbasen benutzt. Im Folgenden soll nun gezeigt werden, wie sich das Prinzip der Gaußelimination, die Elimination der Variablen, auf den allgemeinen Fall übertragen lässt.

Das prinzipielle Vorgehen dabei ist, sogenannte Eliminationsideale zu bestimmen:

Definition 3.3.1
Sei $\mathfrak{a} = (f_1, \ldots, f_m) \subseteq K[X]$ *ein Ideal und* $k \in \{0, \ldots, n\}$. *Dann ist das k-te* ***Eliminationsideal*** $\mathfrak{a}_k$ *dasjenige Ideal in* $K[x_{k+1}, \ldots, x_n]$, *das definiert wird durch*

$$\mathfrak{a}_k := \mathfrak{a} \cap K[x_{k+1}, \ldots, x_n].$$

Analog zum linearen Fall, sollte es also das Ziel sein, eine Basis des gegebenen Ideals zu bestimmen, so dass gewisse Teilmengen davon Basen der Eliminationsideale sind. Dann wäre es, analog zum Rückwärtseinsetzen, möglich, das Bestimmen der Lösungen auf das Lösen univariater Systeme zurückzuführen.
Das folgende Eliminationstheorem zeigt nun, dass *lexikographische Gröbnerbasen* eben diese Anforderungen erfüllen:

Theorem 3.3.2 (Eliminationstheorem)
Sei $G = \{g_1, \ldots, g_m\}$ *eine Gröbnerbasis des Ideals* $\mathfrak{a} \subseteq K[X]$ *bezüglich der lexikographischen Termordnung mit* $x_1 > x_2 > \ldots > x_n$ *und* $k \in \{0, \ldots, n\}$. *Dann bildet die Menge*

$$G_k := G \cap K[x_{k+1}, \ldots, x_n]$$

aller Polynome aus G, die nur Terme in den Variablen $x_{k+1}, \ldots, x_n$ *enthalten, eine Gröbnerbasis des k-ten Eliminationsideals* $\mathfrak{a}_k$.
Beweis:
Nach der Definition einer Gröbnerbasis genügt es zu zeigen, dass der Leitterm jedes $f \in \mathfrak{a}_k$ teilbar ist durch $lt(g)$ für ein $g \in G_k$.
Sei dazu $f \in \mathfrak{a}_k \setminus \{0\}$. Dann ist auch $f \in \mathfrak{a}$ und da G eine Gröbnerbasis für $\mathfrak{a}$ ist, gilt weiter $lt(g_i) \mid lt(f)$ für ein $i \in \{1, \ldots, m\}$. Da $f \in K[x_{k+1}, \ldots, x_n]$ ist, enthält $lt(f)$ und damit insbesondere auch $lt(g_i)$ als Teiler keine der Variablen $x_1, \ldots, x_k$. Zudem ist $lt(g_i)$ aber der Leitterm von g_i bezüglich der lexikographischen Termordnung mit $x_1 > \ldots > x_n$. Damit enthält auch g_i keine der Variablen $x_1, \ldots, x_k$, da jeder Term, der eine dieser Variablen enthält, in dieser Termordnung größer wäre als der Leitterm $lt(g_i)$.
Also ist $g_i \in G \cap K[x_{k+1}, \ldots, x_n] = G_k$ und damit $lt(f)$ für jedes $f \in \mathfrak{a}_k$ teilbar durch ein $lt(g)$ für ein $g \in G_k$. ■

Für alle Lösungen $a = (a_1, \ldots, a_n) \in V_L(\mathfrak{a})$ des Gesamtsystems gilt, dass die sogenannten Teillösungen $(a_{k+1}, \ldots, a_n)$ gerade Lösungen der durch die Eliminationsideale $\mathfrak{a}_k$ beschriebenen Systeme sind, d.h. $(a_{k+1}, \ldots, a_n) \in V_L(\mathfrak{a}_k)$. Insbesondere liefert jede Teillösung $(a_k, \ldots, a_n) \in V_L(\mathfrak{a}_{k-1})$ eine um eine Stelle kürzere Teillösung $(a_{k+1}, \ldots, a_n) \in V_L(\mathfrak{a}_k)$. Zusammen mit dem Eliminationstheorem legt dies analog zum Rückwärtseinsetzen der Gaußelimination folgendes Vorgehen zur Bestimmung der Lösungen nahe:

Zu einer bekannten Teillösung $(a_{k+1}, \ldots, a_n) \in V_L(\mathfrak{a}_k)$ werden in der Basis G_{k-1} von $\mathfrak{a}_{k-1}$ für $i = k+1, \ldots, n$ alle x_i durch die a_i der Teillösung substituiert. Dies ergibt eine Basis des univariaten Ideals, dessen Varietät über L gerade alle möglichen Werte ξ enthält, die $(a_{k+1}, \ldots, a_n)$ zu einer Teillösung $(\xi, a_{k+1}, \ldots, a_n) \in V_L(\mathfrak{a}_{k-1})$ ergänzen.
Sind dies jeweils nur endlich viele, kann auf diese Weise also, ausgehend von den Lösungen des ebenfalls univariaten Eliminationsideals $\mathfrak{a}_{n-1,}$ sukzessive die komplette Lösungsgesamtheit $V_L(\mathfrak{a}) = V_L(\mathfrak{a}_0)$ bestimmt werden.

Bei diesem Vorgehen bleiben noch zwei Fragen offen:

- Welche Teillösungen aus $V_L(\mathfrak{a}_k)$ lassen sich überhaupt zu Teillösungen aus $V_L(\mathfrak{a}_{k-1})$ ergänzen?
- In welchen Fällen enthält eine der Varietäten $V_L(\mathfrak{a}_k)$ unendlich viele Elemente, die somit nicht alle, wie oben beschrieben, in endlicher Zeit weiterverfolgt werden können?

Ein Kriterium dafür, welche Teillösungen sich fortsetzen lassen, liefert — zumindest über algebraisch abgeschlossenen Körpern — das folgende Erweiterungstheorem:

Theorem 3.3.3 (Erweiterungstheorem)
Sei K algebraisch abgeschlossen, $\mathfrak{a} = (g_1, \ldots, g_m) \subset K[X]$ und existieren zudem Polynome $\widetilde{g_1}, \ldots, \widetilde{g_m} \in K[x_2, \ldots, x_n]$, so dass sich g_i als Polynom in x_1 betrachtet als

$$g_i(x_1, \ldots, x_n) = \widetilde{g_i}(x_2, \ldots, x_n) \cdot x_1^{N_i} + \textit{Monome niedrigeren Grades in } x_1$$

mit $N_i \geq 0$ und $\widetilde{g_i} \neq 0$ (außer für $g_i = 0$) schreiben lässt. Sei weiter $(a_2, \ldots, a_n) \in V(\mathfrak{a}_1)$ eine Lösung des ersten Eliminationsideals $\mathfrak{a}_1$. Dann gilt:
Falls $(a_2, \ldots, a_n) \notin V(\widetilde{g_1}, \ldots, \widetilde{g_m})$ ist, dann existiert ein $a_1 \in K$, so dass

$$(a_1, a_2, \ldots, a_n) \in V(\mathfrak{a}).$$

Ein Beweis für den Fall $K = \mathbb{C}$ ist in [CLO92] gegeben. Dieser lässt sich aber, wie ebenfalls in [CLO92] erwähnt wird, direkt auf beliebige, algebraisch abgeschlossene Körper verallgemeinern.

Das Kriterium ist in diesem Theorem nur für den letzten Schritt, die Erweiterung von $V(\mathfrak{a}_1)$ auf $V(\mathfrak{a})$ formuliert. Aber natürlich kann für $\mathfrak{a}$ ein beliebiges der Eliminationsideale $\mathfrak{a}_k$ eingesetzt werden, so dass dieses Theorem zumindest für die einzelnen Schritte von $V(\mathfrak{a}_k)$ nach $V(\mathfrak{a}_{k-1})$ ein Kriterium liefert, welche Lösungen sich fortsetzen lassen.

Im Folgenden wird nun gezeigt, dass es auf die beiden obigen Fragen zumindest im Falle nulldimensionaler Ideale über algebraisch abgeschlossenen Körpern eine einfache Antwort gibt. Dazu dient folgende Charakterisierung nulldimensionaler Ideale:

Satz 3.3.4

Sei K algebraisch abgeschlossen und $\mathfrak{a} \subseteq K[X]$ ein Ideal. Dann sind äquivalent:

i) $V(\mathfrak{a})$ ist endlich,

ii) Für jede Gröbnerbasis G von $\mathfrak{a}$ gilt:
Für alle $i = 1, \ldots, n$ gibt es ein $d_i \in \mathbb{N}_0$ und ein $g_i \in G$, so dass $x_i^{d_i} = lt(g_i)$ ist,

iii) Für alle $i = 1, \ldots, n$ ist $\mathfrak{a} \cap K[x_i] \neq (0)$.

Beweis:

i) $\Rightarrow$ *ii)* : Sei G eine Gröbnerbasis von $\mathfrak{a}$. Für $V(\mathfrak{a}) = \emptyset$ liefert der schwache Hilbertsche Nullstellensatz $1 \in \mathfrak{a}$. Nach Lemma 3.1.17 existiert damit ein $g \in G$ mit $lt(g) = 1$, d.h. $d_i = 0$ für alle i liefert die Behauptung. Sei nun $V(\mathfrak{a}) \neq \emptyset$ und $i \in \{1, \ldots, n\}$. Dann ist auch $V_i := \{a \in K \mid \exists \xi = (\xi_1, \ldots, \xi_n) \in V(\mathfrak{a}) : \ a = \xi_i\}$ eine endliche, nichtleere Menge. Damit sei das Polynom $f \in K[X]$ definiert als

$$f := \prod_{a \in V_i} (x_i - a).$$

Nach Konstruktion ist $f(\xi) = 0$ für alle $\xi \in V(\mathfrak{a})$, also $f \in I(V(\mathfrak{a}))$. Da K algebraisch abgeschlossen ist, existiert damit nach dem Hilbertschen Nullstellensatz ein $d \in \mathbb{N}$, so dass $f^d \in \mathfrak{a}$ ist. Insbesondere ist also $lt\left(f^d\right) = x_i^{d \cdot |V_i|} \in (lt(\mathfrak{a}))$ und wird somit nach Lemma 3.1.17 von einem $lt(g)$ mit $g \in G$ geteilt, d.h. es ist $lt(g) = x_i^{d_i}$ mit $d_i \leq d|V_i|$.
ii) $\Rightarrow$ *iii)* : Für $i \in \{1, \ldots, n\}$ sei G eine Gröbnerbasis zur lexikographischen Termordnung mit $x_1 > \ldots > x_{i-1} > x_{i+1} > \ldots > x_n > x_i$. Nach *ii)* existiert ein $g \in G$ mit $lt(g) = x_i^{d_i}$ für ein $d_i \in \mathbb{N}_0$. Also ist $g \neq 0$ und vor allem auch $g \in K[x_i]$ ein univariates Polynom. Wegen $g \in \mathfrak{a}$ ist also $\mathfrak{a} \cap K[x_i] \neq (0)$.
iii) $\Rightarrow$ *i)* : Für $i = 1, \ldots, n$ seien $f_i \in (\mathfrak{a} \cap K[x_i]) \setminus \{0\}$ und $V_i := \{a \in K \mid f_i(a) = 0\}$ die zugehörigen Varietäten in K. Da wegen $f_i \in \mathfrak{a}$ aber für $\xi = (\xi_1, \ldots, \xi_n) \in V(\mathfrak{a})$ auch $f_i(\xi_i) = 0$ sein muss, gilt $V(\mathfrak{a}) \subseteq V_1 \times \ldots \times V_n$, d.h. wegen $|V_i| \leq grad(f_i)$ ist auch $V(\mathfrak{a})$ endlich. ■

Dies führt zu folgender Antwort auf die beiden obigen Fragen:

Satz 3.3.5

Sei K algebraisch abgeschlossen und $\mathfrak{a} \subseteq K[X]$ ein nulldimensionales Ideal. Dann lässt sich jede Teillösung $(a_{k+1}, \ldots, a_n) \in V(\mathfrak{a}_k)$ zu einer Lösung des Gesamtsystems $(\xi_1, \ldots, \xi_k, a_{k+1}, \ldots, a_n) \in V(\mathfrak{a})$ erweitern.
Insbesondere sind die $V(\mathfrak{a}_k)$ alle endlich.

Beweis:

Seien G und G_k Gröbnerbasen bezüglich der lexikographischen Termordnung von $\mathfrak{a}$ bzw. $\mathfrak{a}_k$ wie im Eliminationstheorem 3.3.2. Da $\mathfrak{a}$ nulldimensional, also $V(\mathfrak{a})$ endlich ist, existiert

nach Satz 3.3.4 ein $d_k \in \mathbb{N}_0$ und $g \in G$ mit $lt(g) = x_k^{d_k}$. Aufgrund der lexikographischen Termordnung ist also $g \in K[x_k, \ldots, x_n]$, d.h. auch $g \in G_{k-1}$ und als Polynom in x_k aufgefasst, lässt sich g schreiben als

$$g = c \cdot x_k^{d_k} + \text{Monome niedrigeren Grades bezüglich } x_k \quad \text{mit } c \in K \setminus \{0\}.$$

Nun kann das Erweiterungstheorem 3.3.3 auf das Eliminationsideal $\mathfrak{a}_{k-1} \subseteq K[X]$ und die Gröbnerbasis G_{k-1} angewendet werden: Nach der obigen Darstellung für g ist unter den $\widetilde{g_i}$ im Erweiterungstheorem ein konstantes Polynom, nämlich das zu $g \in G_{k-1}$ gehörige $\widetilde{g} = c \neq 0$. Die Varietät $V(\widetilde{g_1}, \ldots, \widetilde{g_m})$ muss also leer sein und damit lässt sich nach dem Erweiterungstheorem jede Teillösung $(a_{k+1}, \ldots, a_n) \in V(\mathfrak{a}_k)$ zu einer Teillösung $(a_k, \ldots, a_n) \in V(\mathfrak{a}_{k-1})$ erweitern. Induktiv folgt die Behauptung. ■

Zusammenfassend können die Lösungsgesamtheiten von Polynomsystemen, deren Varietät über dem algebraischen Abschluss endlich ist, also folgendermaßen bestimmt werden:

Algorithmus 3.3.6 (Lösen von Gleichungssystemen)
Eingabe:

Ideal $\mathfrak{a} = (f_1, \ldots, f_m) \subseteq K[X]$

Ausgabe:

Lösungsgesamtheit $V(\mathfrak{a})$

1. Bestimme mit Hilfe des Buchberger-Algorithmus eine minimale, lexikographische Gröbnerbasis G von $\mathfrak{a}$ und damit auch gleichzeitig Gröbnerbasen G_k zu den Eliminationsidealen $\mathfrak{a}_k$.

2. Bestimme die Lösungsgesamtheit $V(\mathfrak{a}_{n-1})$ des letzten Eliminationsideals $\mathfrak{a}_{n-1}$ durch Faktorisierung des univariaten Polynoms in G_{n-1}, das $\mathfrak{a}_{n-1}$ erzeugt.

3. Für k von $n-1$ bis 1 wiederhole:
   Für alle bisher berechneten Teillösungen $(a_{k+1}, \ldots, a_n) \in V(\mathfrak{a}_k)$ mache das Folgende: Substituiere $x_j = a_j$ für $j = k+1, \ldots, n$ in der Basis G_k. Dies liefert ein univariates System, dessen Lösungen a_k wie im univariaten Fall bestimmt werden können. Das ergibt neue Teillösungen $(a_k, \ldots, a_n) \in V(\mathfrak{a}_{k-1})$.

4. Die so berechneten (Teil-)Lösungen $(a_1, \ldots, a_n) \in V(\mathfrak{a}_0)$ bilden die Lösungsgesamtheit $V(\mathfrak{a})$.

Nach einem Theorem von Gianni und Kalkbrenner (siehe [Laz92]) lässt sich der dritte Schritt noch weiter vereinfachen, so dass die Substitution direkt das Polynom liefert, dessen Nullstellen die Ergänzungen der Teillösung ergeben.

3.3.1 FGLM-Algorithmus

Im Abschnitt über die Komplexität des Buchberger-Algorithmus wurde schon erwähnt, dass in der Praxis die Berechnung lexikographischer Gröbnerbasen im Allgemeinen schwieriger ist als die Berechnung beispielsweise graduierter Gröbnerbasen. Im vorigen Abschnitt wurde aber gezeigt, dass gerade die lexikographischen Gröbnerbasen für das Lösen von polynomialen Gleichungssystemen besonders hilfreich sind. Deshalb haben Faugère, Gianni, Lazard und Mora in [FGLM93] ein Verfahren vorgestellt, mit dem im Fall eines nulldimensionalen Ideals Gröbnerbasen von einer Termordnung in eine andere Termordnung umgerechnet werden können und zwar in einer bezüglich der Anzahl der gemeinsamen Nullstellen polynomiellen Laufzeit.

Dieser sogenannte FGLM-Algorithmus basiert im Wesentlichen auf der von den Termordnungen unabhängigen Vektorraumstruktur von $K[X]/\mathfrak{a}$ und ihrem von einer gewählten Termordnung abhängigen Zusammenhang mit dem Vektorraum $\langle \mathcal{N} \rangle_K$.
Es ist leicht nachzurechnen (wie es beispielsweise in [CLO92] durchgeführt wird), dass Folgendes gilt:

Satz 3.3.7
Sei $\mathfrak{a} \subseteq K[X]$ ein Ideal und $\mathcal{N}$ die Normalmenge von $\mathfrak{a}$ bezüglich einer Termordnung $<$. Dann ist

$$\phi_< : \begin{array}{ccc} K[X]/\mathfrak{a} & \to & \langle \mathcal{N} \rangle_K \\ f + \mathfrak{a} & \mapsto & NF(f, \mathfrak{a}) \end{array}$$

ein K-Isomorphismus.

Im Folgenden werden nun die wesentlichen Ideen des FGLM-Algorithmus vorgestellt.

Seien zwei Termordnungen $<_1$ und $<_2$ und eine Gröbnerbasis G_1 bezüglich $<_1$ eines Ideals $\mathfrak{a}$ gegeben.
Da die Normalmenge mit Hilfe des Leittermideals definiert ist, gibt es zu $\mathfrak{a}$ zwei im Allgemeinen verschiedene Normalmengen $\mathcal{N}_1$ und $\mathcal{N}_2$ bezüglich der verschiedenen Termordnungen $<_1$ und $<_2$. Durch die Gröbnerbasis G_1 bezüglich der ersten Termordnung kann effizient mit Hilfe des Reduktionsalgorithmus zu jedem $f \in K[X]$ die Normalform $NF_1(f, \mathfrak{a}) \in \langle \mathcal{N}_1 \rangle_K$ bezüglich $<_1$ bestimmt werden. Die Gröbnerbasis G_1 liefert also den Isomorphismus $\phi_{<_1}$ und damit die Möglichkeit, in $K[X]/\mathfrak{a}$ zu rechnen.

Um nun eine Gröbnerbasis $G_2 = \{g_1, \ldots, g_m\}$ bezüglich der anderen Termordnung $<_2$ zu erhalten, ist es nötig, Polynome $g_1, \ldots, g_m \in \mathfrak{a}$ zu finden, deren Leitterme bezüglich $<_2$ das Leittermideal $(lt_2(\mathfrak{a}))$ erzeugen. Dazu ist es hilfreich, zunächst festzustellen welche Terme in $\mathcal{N}_2$, und damit gerade nicht in $(lt_2(\mathfrak{a}))$ liegen. Die Grundlage dafür ist das folgende Lemma:

Lemma 3.3.8

Seien $\mathfrak{a}$, $<_1$ und $<_2$ wie oben beschrieben gegeben. Dann sind für $t \in \mathcal{T}$ äquivalent:

i) $t \in \mathcal{N}_2$,

ii) $NF_2(t, \mathfrak{a}) = t$,

iii) $\{NF_2(t, \mathfrak{a})\} \cup \{t' \in \mathcal{N}_2 \mid t' <_2 t\} =: T_2$ *ist K-linear unabhängig,*

iv) $\{NF_1(t, \mathfrak{a})\} \cup \{NF_1(t', \mathfrak{a}) \mid t' \in \mathcal{N}_2,\ t' <_2 t\} =: T_1$ *ist K-linear unabhängig.*

Beweis:

i) $\Leftrightarrow$ *ii)* ist klar nach Satz 3.1.18 und der Eindeutigkeit der Normalform.

ii) $\Rightarrow$ *iii)* ist ebenfalls klar, da für $NF_2(t, \mathfrak{a}) = t$ die gesamte Menge T_2 nur aus unterschiedlichen Termen besteht.

iii) $\Rightarrow$ *ii)*: Angenommen, es wäre $NF_2(t, \mathfrak{a}) \neq t$. Wegen $NF_2(t, \mathfrak{a}) = Reduktion(f, G)$ enthält $NF_2(t, \mathfrak{a})$ dann nur Monome die bezüglich $<_2$ kleiner sind als t.
Wegen $NF_2(t, \mathfrak{a}) \in \langle \mathcal{N}_2 \rangle_K$ wäre also $NF_2(t, \mathfrak{a}) \in \langle \{t' \in \mathcal{N}_2 \mid t' < t\} \rangle_K$ im Widerspruch zur linearen Unabhängigkeit.

iii) $\Leftrightarrow$ *iv)* Die durch NF_1 und NF_2 gegebenen Isomorphismen $\phi_{<_1}$ und $\phi_{<_2}$ zwischen $K[X]/\mathfrak{a}$ und $\langle \mathcal{N}_1 \rangle_K$ bzw. $\langle \mathcal{N}_2 \rangle_K$ liefern einen Isomorphismus $\phi := \phi_{<_2} \circ \phi_{<_1}^{-1}$ zwischen $\langle \mathcal{N}_1 \rangle_K$ und $\langle \mathcal{N}_2 \rangle_K$ mit $\phi(NF_1(f, \mathfrak{a})) = NF_2(f, \mathfrak{a})$ für alle $f \in K[X]$. Insbesondere ist wegen $NF_2(t', \mathfrak{a}) = t'$ für $t' \in \mathcal{N}_2$ dann $\phi(T_1) = T_2$ und daraus folgt die Behauptung. ■

Ist für $t_0 \in \mathcal{T}$ die Menge $B := \{t' \in \mathcal{N}_2 \mid t' <_2 t_0\}$ bekannt, so kann mit Hilfe der Gröbnerbasis G_1 die Menge T_1 berechnet und ihre lineare Unabhängigkeit überprüft werden, um festzustellen, ob $t_0 \in \mathcal{N}_2$ ist.
Im FGLM-Algorithmus geschieht dies so, dass die Terme $t_0 \in \mathcal{T}$ in aufsteigender Reihenfolge bezüglich $<_2$ durchlaufen werden. Dadurch ist zum aktuellen $t_0 \in \mathcal{T}$ immer schon die zugehörige Menge B bekannt, so dass auf diese Weise sukzessive die Menge $\mathcal{N}_2$ aufgebaut werden kann. Wird dabei ein Term t_0 gefunden, der nicht in $\mathcal{N}_2$ liegt, ist klar, dass t_0 und damit auch alle Vielfachen $t \cdot t_0$ mit $t \in \mathcal{T}$ im Leittermideal $(lt_2(\mathfrak{a}))$ liegen und im weiteren Verlauf nicht mehr betrachtet werden müssen.

Zudem muss t_0 durch den Leitterm eines Elements der zu bestimmenden Gröbnerbasis G_2 teilbar sein. Dies wird sichergestellt, indem immer, wenn ein solches t_0 gefunden wird, ein Polynom aus $\mathfrak{a}$ zu G_2 hinzugefügt wird, das t_0 als Leitterm besitzt. Dieses kann leicht folgendermaßen gefunden werden:
Aus Lemma 3.3.8 folgt, dass für $t_0 \notin \mathcal{N}_2$ gewisse $c_{t'} \in K$ mit

$$NF_1(t_0, \mathfrak{a}) = \sum_{t' \in B} c_{t'} \cdot NF_1(t', \mathfrak{a}).$$

existieren. Damit ist in $K[X]/\mathfrak{a}$ aber

$$\left(t_0 - \sum_{t' \in B} c_{t'} \cdot t'\right) + \mathfrak{a} = \phi_{<_1}^{-1}\left(NF_1\left(t_0, \mathfrak{a}\right) - \sum_{t' \in B} c_{t'} \cdot NF_1\left(t', \mathfrak{a}\right)\right) = \phi_{<_1}^{-1}\left(0\right) = 0 + \mathfrak{a},$$

also $g := t_0 - \sum_{t' \in B} c_{t'} \cdot t' \in \mathfrak{a}$ und wegen $t' \in B \Rightarrow t' <_2 t_0$ folgt $lt_2\left(g\right) = t_0$. Deshalb kann g zu G_2 hinzugefügt werden und stellt sicher, dass alle bisher in $\left(lt_2\left(\mathfrak{a}\right)\right)$ gefundenen Terme durch den Leitterm mindestens eines Elements aus G_2 teilbar sind.

Damit bleibt nur noch zu klären, warum es ausreicht, nur endlich viele der Terme aus $\mathcal{T}$ zu betrachten: Für jeden Term t_0, der im Algorithmus wie oben beschrieben betrachtet wird, gilt zum Zeitpunkt der Betrachtung das Folgende: Entweder ist t_0 in $\mathcal{N}_2$ enthalten oder t_0 ist der kleinste Term außerhalb von $\mathcal{N}_2$, der nicht in $\left(lt_2\left(g\right) \mid g \in G_2 \right)$ liegt. Da Normalmengen von nulldimensionalen Idealen endlich sind, kann es von der ersten Art nur endlich viele geben. Ist t_0 von der zweiten Art, wird in diesem Schritt ein Polynom g mit $lt_2\left(g\right) = t_0$ zu G_2 hinzugefügt, so dass das Ideal $\left(lt_2\left(g\right) \mid g \in G_2 \right)$ echt größer wird. Da aber $K\left[X\right]$ noethersch ist, kann auch dieser Fall nur endlich oft eintreten, d.h. es werden nur endlich viele Terme t_0 betrachtet.

Zusammengefasst hat der FGLM-Algorithmus also folgende Gestalt:

Algorithmus 3.3.9 (FGLM)

Eingabe:

Termordnungen $<_1$ und $<_2$ von $\mathcal{T}$,

Gröbnerbasis G_1 des nulldimensionalen Ideals $\mathfrak{a}$ bezüglich $<_1$

Ausgabe:

Gröbnerbasis $G_2 = \{g_1, \ldots, g_m\}$ von $\mathfrak{a}$ bezüglich der Termordnung $<_2$

FGLM$(<_1, <_2, G_1)$:

$T := \mathcal{T}$, $G_2 := \emptyset$, $B := \emptyset$ ## in B werden die Terme aus $\mathcal{N}_2$ gesammelt

Wiederhole

$t_0 := \min_{<_2} T$

$T := T \setminus \{t_0\}$

Wenn $NF_1\left(t_0, \mathfrak{a}\right) \in \langle\, NF_1\left(b, \mathfrak{a}\right) \mid b \in B \,\rangle_K$

dann ## t_0 gehört nicht zu $\mathcal{N}_2$

Bestimme $c_b \in K$ mit $NF_1\left(t_0, \mathfrak{a}\right) = \sum_{b \in B} c_b \cdot NF_1\left(b, \mathfrak{a}\right)$

$G_2 := G_2 \cup \left\{t_0 - \sum_{b \in B} c_b \cdot b\right\}$

$T := T \setminus \{t \cdot t_0 \mid t \in \mathcal{T}\}$ ## die Vielfachen von t_0 brauchen ## nicht mehr betrachtet zu werden

sonst ## t_0 gehört zu $\mathcal{N}_2$

$B := B \cup \{t_0\}$

bis $T = \emptyset$

return G_2

In dieser Form kann der Algorithmus allerdings nicht in die Praxis umgesetzt werden, da die Menge T nicht endlich und somit im Allgemeinen nicht speicherbar ist. Zudem ist die Komplexität der Berechnung der Normalform durch den Reduktionsalgorithmus nur schwer kontrollierbar.
In [FGLM93] werden deshalb bessere Möglichkeiten vorgestellt, den nächsten zu betrachtenden Term zu ermitteln und die Normalformen zu berechnen. Dies führt zu folgenden Abschätzungen der Komplexität des FGLM-Algorithmus:

Theorem 3.3.10
Sei $\mathfrak{a} \subseteq K[X]$ ein nulldimensionales Ideal, G_1 eine Gröbnerbasis von $\mathfrak{a}$ bezüglich einer Termordnung $<_1$ und $D := \dim_K (K[X]/\mathfrak{a})$. Dann kann eine Gröbnerbasis G_2 bezüglich einer weiteren Termordnung $<_2$ mit $O(nD^3)$ arithmetischen Operationen in K konstruiert werden.

Der exakte Algorithmus mit dieser Komplexität und der vollständige Beweis stehen in [FGLM93].

3.4 Anwendung von Gröbnerbasistechniken auf HFE

In diesem Abschnitt wird untersucht, wie die bislang vorgestellten Techniken zum Lösen von Gleichungssystemen auf die bei HFE auftretenden Gleichungssysteme angewendet werden können.
Dazu wird in Abschnitt 3.4.1 zunächst beschrieben, was ein HFE-System von anderen Systemen unterscheidet und warum diese Unterschiede Anlass zu der Hoffnung geben, dass HFE-Systeme durchschnittlich effizienter zu lösen sein könnten als allgemeine Systeme. Anschließend werden in Abschnitt 3.4.2 dann die Ergebnisse von einigen, eigenen Simulationen vorgestellt, in denen praktisch untersucht wurde, wieviel Zeit für das Lösen eines HFE-Systems mit Gröbnerbasistechniken in Abhängigkeit von den Parametern n, q und d benötigt wird.

3.4.1 Theoretische Betrachtungen

Gegeben sei ein öffentlicher Schlüssel $P : K^n \to K^n$ eines HFE-Systems mit $K = \mathbb{F}_q$, dem wie üblich eine (unbekannte) versteckte Funktion zugrunde liegt, und ein Chiffretext $y \in K^n$, der entschlüsselt werden soll. Es gilt also, eine Lösung (bzw. möglichst alle Lösungen) $x \in K^n$ des Systems

$$\begin{aligned} p_1(x_1, \ldots, x_n) &= y_1 \\ \vdots \qquad\qquad\quad &\vdots \quad\; \vdots \\ p_n(x_1, \ldots, x_n) &= y_n \end{aligned}$$

zu finden, wobei $(p_1, \ldots, p_n) \in \mathcal{M}_{n,2}$ die multivariate Darstellung der öffentlichen Funktion P beschreibt. Es sei zudem ohne Einschränkung $y = 0$ (betrachte sonst im Folgenden $p'_i := p_i - y_i$ anstelle der p_i).
Dann besteht das allgemeine, oben beschriebene Verfahren zur Lösung eines solchen Systems darin, zum Ideal

$$\mathfrak{a} := (p_1, \ldots, p_n)$$

etwa mit Hilfe des FGLM-Algorithmus eine lexikographische Gröbnerbasis zu berechnen und daraus die Lösungen abzulesen.

Allerdings ist die worst case Komplexität des Buchberger-Algorithmus (vgl. Abschnitt 3.2.2) so groß, dass für allgemeine Systeme mit mehr als schätzungsweise 10 Variablen davon ausgegangen werden kann, dass für diese in akzeptabler Zeit meist keine Gröbnerbasis berechnet werden kann.
Die Hoffnung, dass HFE-Systeme — bei denen üblicherweise deutlich mehr als 10 Variablen verwendet werden — vielleicht schneller lösbar sind als allgemeine, polynomiale Gleichungssysteme mit entsprechender Variablenanzahl, stützt sich auf folgende, spezielle Eigenschaften von HFE-Systemen:

- HFE arbeitet über **kleinen**, **endlichen** Körpern $K = \mathbb{F}_q$.
- HFE-Systeme bestehen nur aus **quadratischen** Polynomen $p_1, \ldots, p_n$.
- Es werden nur **Lösungen im Grundkörper** (nicht im algebraischen Abschluss) gesucht.

Vor allem der letzte Punkt beschreibt einen großen Vorteil gegenüber der in den vorherigen Abschnitten beschriebenen, allgemeinen Lösung von polynomialen Gleichungssystemen. Denn, wie in Abschnitt 3.3 gesehen, enthält eine lexikographische Gröbnerbasis eines Ideals $\mathfrak{a}$ immer schon die Informationen, die nötig sind, um effizient alle Elemente aus $V_{\overline{K}}(\mathfrak{a})$ zu berechnen. Im Allgemeinen bilden aber die hier gesuchten Lösungen $V_K(\mathfrak{a})$ im Grundkörper nur eine kleine Teilmenge von $V_{\overline{K}}(\mathfrak{a})$.

Um bei HFE-Systemen (oder auch bei beliebigen, polynomialen Gleichungssystemen über endlichen Körpern, bei denen nur Lösungen im Grundkörper gesucht werden) die Menge der durch die Gröbnerbasis „berechneten“ Lösungen auf die wirklich gesuchten einzuschränken, kann das betrachtete Ideal geeignet erweitert und $V_{\overline{K}}(\mathfrak{a})$ so entsprechend eingeschränkt werden.
Da für alle Elemente $b \in \mathbb{F}_q$ eines endlichen Körpers mit q Elementen $b^q = b$ gilt, können Polynome der Form ${x_i}^q - x_i$ problemlos zu $\mathfrak{a}$ hinzugenommen werden, ohne dadurch Elemente aus $V_K(\mathfrak{a})$ zu verlieren. Genauer gesagt gilt sogar folgender Satz:

Lemma 3.4.1

Sei $\mathfrak{a} \subseteq K[x_1, \ldots, x_n]$ *ein Ideal über einem endlichen Körper* $K = \mathbb{F}_q$. *Dann gilt*

$$V_K(\mathfrak{a}) \quad = \quad V_{\overline{K}}(\mathfrak{a} + ({x_1}^q - x_1, \ldots, {x_n}^q - x_n)).$$

Beweis:

Es gilt

$$\begin{aligned} V_{\overline{K}}(\mathfrak{a} + ({x_1}^q - x_1, \ldots, {x_n}^q - x_n)) &= V_{\overline{K}}(\mathfrak{a}) \cap \underbrace{V_{\overline{K}}({x_1}^q - x_1, \ldots, {x_n}^q - x_n)}_{=K^n} \\ &= V_{\overline{K}}(\mathfrak{a}) \cap K^n \\ &= V_K(\mathfrak{a}). \end{aligned}$$

■

Um Lösungen von $P(x) = y$ in K^n zu finden, ist es also sinnvoll, statt $\mathfrak{a}$ das Ideal

$$\mathfrak{a}' := (p_1, \ldots, p_n, {x_1}^q - x_1, \ldots, {x_n}^q - x_n)$$

zu betrachten. Die Hoffnung dabei ist natürlich, dass durch die Beschränkung der Informationen, die die berechnete Gröbnerbasis liefert (nämlich auf Lösungen in K^n statt in $\overline{K}^n$), auch der Aufwand, sie zu berechnen, entsprechend eingeschränkt wird.

Unterstützt wird diese Hoffnung noch von folgender Tatsache, die Mayr in [May97] im Zusammenhang mit der Verallgemeinerung der $\mathcal{EXPSPACE}$-Vollständigkeit des IM-Problems (vgl. Abschnitt 3.2.2) auf beliebige, unendliche Körper erwähnt: Um im Falle unendlicher Körper von positiver Charakteristik ebenfalls die exponentielle, untere Schranke beweisen zu können, müssen nämlich gerade Ideale, die Polynome der Form ${x_i}^q - x_i$ enthalten, ausgeschlossen werden.

Ein weiterer Vorteil des Ideals $\mathfrak{a}'$ gegenüber $\mathfrak{a}$ ist, dass $V_{\overline{K}}(\mathfrak{a}') = V_K(\mathfrak{a}') \subseteq K^n$ endlich, also $\mathfrak{a}'$ nulldimensional ist. Nach den in Abschnitt 3.2.2 dargestellten Zusammenhängen ist die Berechnung einer Gröbnerbasis von $\mathfrak{a}'$ also im worst case „nur“ einfach exponentiell und zudem ist auch der FGLM-Algorithmus aus Abschnitt 3.3.1 durchführbar.

Diese einfach exponentielle Schranke lässt sich im speziellen Fall von $\mathfrak{a}'$ auch noch auf folgende Art nachvollziehen:
Es ist leicht zu sehen, dass für Normalformen $NF(f, \mathfrak{a}')$ bezüglich graduierter Termordnungen immer

$$grad\left(NF\left(f, \mathfrak{a}'\right)\right) \quad \leq \quad (q-1)\,n$$

gilt. Denn für graduierte Termordnungen gilt $grad(f) = grad(lt(f))$ und jedes Monom, das einen Faktor ${x_i}^l$ mit $l \geq q$ enthält, kann bezüglich $\mathfrak{a}'$ natürlich mittels ${x_i}^q - x_i$ noch reduziert werden.

Bei geschickter Implementierung hat diese Beobachtung zur Folge, dass Polynome mit einem Grad $> (q-1)\,n$ während des Buchberger-Algorithmus nicht betrachtet werden müssen. Für diese (sogenannte gradbeschränkte) Variante des Buchberger-Algorithmus wurde in [BCEMR94] folgende Laufzeitabschätzung gezeigt:

Satz 3.4.2
Sei τ die Anzahl der Terme vom Grad $\leq D$, d.h. $\tau = \binom{n+D}{n}$. Dann kann die Variante des Buchberger-Algorithmus, die alle Rechnungen überspringt, die zu Polynomen mit Grad $> D$ führen, für graduierte Termordnungen in Zeit $O\left(\tau^4\right)$ durchgeführt werden.
Beweisskizze:
Jeder Term vom Grad $\leq D$ kann erhalten werden durch Auswahl von D aus den $n+1$ möglichen Faktoren $1, x_1, x_2, \ldots, x_n$ mit Wiederholung und ohne Beachtung der Reihenfolge, also ist $\tau = \binom{n+1+D-1}{D} = \binom{n+D}{n}$.
Die Zeitkomplexität von $O\left(\tau^4\right)$ wird in [BCEMR94] damit begründet, dass (in den Bezeichnungen von Algorithmus 3.2.1) die Menge P der zu betrachtenden Paare höchstens $O\left(\tau^2\right)$ Elemente enthalten kann und dass für die dafür durchzuführenden Reduktionen auch jeweils nur eine Zeit von $O\left(\tau^2\right)$ benötigt wird. ■

Für $D = (q-1)\,n$ ergibt dies $\tau = \binom{qn}{n} = \prod_{j=0}^{n-1} \frac{qn-j}{n-j} \approx q^n$, also ebenfalls gerade die einfach exponentielle Zeit bezüglich n.

Neben der Möglichkeit, zum betrachteten Ideal noch die Polynome ${x_i}^q - x_i$ hinzuzufügen, gibt es — wie oben schon erwähnt — noch weitere Beobachtungen, die unter Umständen bewirken, dass HFE-Systeme durchschnittlich effizienter lösbar sind, als allgemeine polynomiale Gleichungssysteme:
Die Tatsache etwa, dass HFE über endlichen Körpern arbeitet, bewirkt, dass alle zu speichernden Koeffizienten immer den gleichen Platz belegen. Es können also keine Probleme mit immer weiter wachsenden Darstellungsgenauigkeiten auftreten, wie es etwa beim Bestimmen von Gröbnerbasen über $\mathbb{Q}$ passiert.
Zudem sind die ursprünglichen Polynome $p_1, \ldots, p_n$ immer quadratisch. Der Grad der Eingangspolynome, der auch bei den in Abschnitt 3.2.2 beschriebenen worst case Abschätzungen eine Rolle spielte, ist also immer 2.

Dies sind allerdings alles Feststellungen, die auf jedes beliebige, quadratische Gleichungssystem über endlichen Körpern zutreffen. Die Herkunft des Systems als HFE-System mit einer dahinterliegenden, versteckten Abbildung $\varphi : L \to L$ über einem Erweiterungskörper L von K wurde bislang noch nicht ausgenutzt. Es stellt sich also noch die Frage, ob auch die Existenz dieser versteckten Abbildung die Lösung des Systems weiter vereinfacht:

Eine Vermutung ist, dass zumindest für ziemlich kleine Grade von φ die Lösung wirklich deutlich einfacher wird. Denn ähnlich wie bei der Attacke, mit der Patarin C^* gebrochen hat, hat er auch bei bestimmten geheimen Funktionen φ von vergleichsweise kleinem Grad die Existenz vieler einfacher Relationen zwischen den auftretenden p_i, sogenannter Syzygien, festgestellt (vgl. Kapitel 5). Hinter der Berechnung von Gröbnerbasen mit dem Buchberger-Algorithmus steckt aber im Wesentlichen das Ausnutzen solcher Syzygien (daher auch die Bezeichnung S-Polynom).
Die Vermutung liegt also nahe, dass der Buchberger-Algorithmus deutlich effizienter ist, falls viele einfache Syzygien existieren, was bei HFE-Systemen, denen eine Abbildung φ mit kleinem Grad zugrunde liegt, anscheinend häufig der Fall ist.

3.4.2 Praktische Laufzeituntersuchungen

In diesem Abschnitt werden die Ergebnisse einiger eigener Simulationen vorgestellt und analysiert. In diesen Simulationen wurde untersucht, wie schnell HFE-Systeme (unter anderem im Vergleich zu zufälligen, quadratischen Gleichungssystemen) mit dem Buchberger-Algorithmus in Verbindung mit dem FGLM-Algorithmus lösbar sind. Dabei wurden insbesondere die Überlegungen aus dem letzten Abschnitt berücksichtigt.
Insgesamt wurden dazu ungefähr 96000 Simulationen durchgeführt. In jeder dieser Simulationen wurde zu vorgegebenen $n \in \{4, \ldots, 20\}$ und $q \in \{2, \ldots, 13\}$ ein quadratisches Gleichungssystem mit n Variablen und n Gleichungen über dem Körper $\mathbb{F}_q$ erzeugt und dann eine Gröbnerbasis des zugehörigen Ideals bestimmt. Für die Erzeugung der Gleichungssysteme gab es zwei Möglichkeiten: Entweder wurde zu vorgegebenem Grad d ein HFE-Polynom φ zufällig gewählt und daraus das zugehörige System bestimmt oder das gesamte Gleichungssystem wurde zufällig gewählt, d.h. es wurden n zufällige, quadratische Gleichungen in n Variablen gewählt.
Zudem wurde anhand der berechneten Gröbnerbasis auch immer die Anzahl der Lösungen des Systems bestimmt. Die daraus erhaltene Verteilung der Lösungsanzahlen wurde schon in Abschnitt 2.3.3 ausgewertet.

Die Simulationen wurden auf einem PC mit einem Pentium III-Prozessor mit 750 MHz Taktfrequenz und 128 MB Arbeitsspeicher unter dem Betriebssystem Windows ME durchgeführt. Als Software wurde dazu SINGULAR 2.0.0 für Windows (siehe [GPS01]) mit folgenden Optionen verwendet:
Die Zeitmessung geschah mit der Prozedur `rtimer` unter der Einstellung `-ticks-per-sec=100`, d.h. mit einer Genauigkeit von $\frac{1}{100}$ Sekunde. Zudem wurde die Option `redSB` verwendet, um reduzierte Gröbnerbasen zu berechnen und `noSugar`, um die Verwendung der sogenannten Sugar-Strategie zu vermeiden, da diese anscheinend bei den hier zu lösenden Systemen eine Verlangsamung zur Folge hatte.

Zur Durchführung der Simulationen wurden die beiden Prozeduren `TestGB` und `TestGBZufall` (vgl. Anhang A) verwendet, die beide zu vorgegebenen Parametern n, q und gegebenenfalls d jeweils eine bestimmte Anzahl an Simulationen durchführen.
`TestGB` erzeugt dabei zu den übergebenen Parametern n, q und d für jede Simulation ein HFE-System über dem Körper $\mathbb{F}_q$ mit n Variablen und n Gleichungen, wobei die zugrunde liegende, versteckte Funktion φ zufällig mit $grad(\varphi) \leq d$ gewählt wird. Dieses Gleichungssystem wird als Ideal gespeichert und um die Polynome ${x_i}^q - x_i$ ergänzt. Anschließend wird in dieser Prozedur dann mit Hilfe der Singular-Prozedur `stdfglm`, die eine Implementierung des Buchberger-Algorithmus unter Berücksichtigung des FGLM-Algorithmus darstellt, eine reduzierte Gröbnerbasis des Ideals berechnet und die dafür benötigte Zeit mit Hilfe der Singular-Prozedur `rtimer` gemessen. Die so gemessenen Zeiten werden schließlich noch in eine Datei geschrieben (vgl. Anhang A), wobei jede einzelne Simulation in einem eigenen Datensatz gespeichert wird.
`TestGBZufall` unterscheidet sich von `TestGB` nur dadurch, dass das Gleichungssystem nicht als HFE-System sondern als zufälliges, quadratisches Gleichungssystem über $\mathbb{F}_q$ mit n Variablen und n Gleichungen erzeugt wird.

Auf der beigelegten CD sind sowohl die für die Simulationen verwendeten Prozeduren als auch die gesamten Einzelergebnisse zu finden (vgl. Anhang A). Im Folgenden werden Zusammenfassungen ausgewählter Teile dieser Ergebnisse präsentiert und analysiert.

Zunächst wird untersucht, ob die benötigten Laufzeiten wirklich einfach exponentiell von n abhängen:
Dazu wurden für $q = 2$ und $q = 3$ insgesamt ungefähr 60000 Simulationen für verschiedene Werte von n zwischen 4 und 20 durchgeführt, deren durchschnittlich benötigte Laufzeiten in der folgenden Tabelle zusammengefasst sind (in Sekunden):

	$q=2$			$q=3$			
n	$d=20$	$d=128$	$zufällig$	$d=12$	$d=30$	$d=90$	$zufällig$
4	0,002	0,001	0,002	0,002	0,002	0,003	0,003
5	0,003	0,003	0,002	0,006	0,006	0,006	0,006
6	0,005	0,005	0,004	0,019	0,021	0,018	0,019
7	0,010	0,010	0,009	0,076	0,078	0,078	0,079
8	0,018	0,018	0,019	0,267	0,273	0,272	0,274
9	0,044	0,043	0,044	0,814	1,167	1,168	1,169
10	0,103	0,103	0,102	3,378	5,298	5,328	5,205
11	0,273	0,271	0,273	9,803	29,230	38,112	38,399
12	0,779	0,780	0,789	27,422	81,634	131,974	132,671
13	2,229	2,224	2,265	71,671	243,956	518,868	519,990
14	5,178	8,686	8,705	252,161	1380,614	2269,079	2262,095
15	9,319	20,449	19,754				
16	21,025	54,327	57,307				
17	47,380	142,437	170,194				
18	135,503	358,723	433,198				
19	303,672	876,440	1138,741				
20	689,467	2643,552	3269,263				

Dabei steht „$d = \ldots$“ für Simulationen mit HFE-Systemen, die aus HFE-Polynomen vom Grad d hervorgehen (mit Hilfe der Prozedur `TestGB` getestet) und „zufällig“ für Simulationen mit zufällig gewählten, quadratischen Gleichungssystemen (mit Hilfe der Prozedur `TestGBZufall` getestet).
In dieser Tabelle ist schon deutlich die exponentielle Abhängigkeit der Laufzeiten von n zu erkennen. Eine Regression mit der Zielfunktion $Zeit = c_0 \cdot 2^{c_1 \cdot n}$ ergab folgende Schätzungen für die benötigten Laufzeiten:

$q = 2$	
$d = 20:$	$3,1 \cdot 2^{1,22 \cdot n}\ ms$
$d = 128:$	$1,5 \cdot 2^{1,35 \cdot n}\ ms$
$zufällig:$	$1,2 \cdot 2^{1,38 \cdot n}\ ms$

$q = 3$	
$d = 12:$	$1,9 \cdot 2^{1,70 \cdot n}\ ms$
$d = 30:$	$0,7 \cdot 2^{1,95 \cdot n}\ ms$
$d = 90:$	$0,4 \cdot 2^{2,05 \cdot n}\ ms$
$zufällig:$	$0,5 \cdot 2^{2,05 \cdot n}\ ms$

Dazu ist noch zu erwähnen, dass diese Formeln die gemessenen Zeiten sehr gut abschätzen. Es liegt also anscheinend wirklich eine einfach exponentielle Abhängigkeit vor.
In [CKPS00] wird verwiesen auf Gröbnerbasis-Algorithmen von Faugère, die für $q = 2$ eine asymptotische Komplexität von ungefähr $O\left(2^{2n}\right)$ und für große q in der Praxis ein Komplexität von $O\left(2^{2,7n}\right)$ haben. Die hier gemessenen Laufzeiten liegen also mit $O\left(2^{1,38 \cdot n}\right)$ und $O\left(2^{2,05 \cdot n}\right) \approx O\left(3^{1,29 \cdot n}\right)$ asymptotisch deutlich unter denen von Faugère. Allerdings ist auch der hier verwendete Algorithmus (zumindest asymptotisch) immer noch langsamer als eine vollständige Suche (mit $O\left(q^n\right)$).

An den obigen Ergebnissen ist andeutungsweise schon die in Abschnitt 3.4.1 vermutete Abhängigkeit der Laufzeiten vom Grad d der versteckten Funktion φ zu erkennen. Diese wird im Folgenden anhand einiger spezieller Testreihen näher untersucht.
Etwa für $q = 2$, $n = 12$ ergaben sich folgende durchschnittliche Zeiten (in Sekunden) für verschiedene Werte von d:

d	2	3	4	5	6	8	9	10	12	16	17	18
$Zeit$	0,003	0,058	0,074	0,151	0,143	0,147	0,365	0,353	0,359	0,351	0,786	0,782

Für $q = 3$, $n = 10$ wurden folgende Zeiten gemessen:

d	3	4	6	9	10	12	18	27	28	30	36	54
$Zeit$	0,115	0,894	1,070	1,438	2,703	3,378	3,077	3,771	5,343	5,298	5,347	5,343

In diesen Tabellen sind natürlich nur die Grade d von Polynomen aufgeführt, die für $q = 2$ bzw. $q = 3$ auch tatsächlich als HFE-Polynome auftreten können.
Auffällig bei diesen Ergebnissen sind die deutlichen Sprünge, die immer zwischen $d = q^i$ und $d = q^i + 1$ auftreten. Diese Beobachtung legt nahe, dass die benötigte Zeit nicht direkt vom Grad d sondern im Wesentlichen von $\left\lceil \log_q(d) \right\rceil$ abhängt.

Eine Begründung dafür ist möglicherweise die folgende Feststellung:
In HFE-Polynomen

$$\varphi(x) = \sum_{q^i+q^j \leq d} \beta_{ij} x^{q^i} x^{q^j} + \sum_{q^i \leq d} \alpha_i x^{q^i} + \mu_0$$

kommen im quadratischen Teil $\sum_{q^i+q^j \leq d} \beta_{ij} x^{q^i} x^{q^j}$ nur $\lceil \log_q(d) \rceil$ verschiedene x^{q^i} (nämlich die mit $i \in \{0, \ldots, \lceil \log_q(d) \rceil - 1\}$ vor. Beim Sprung von $d = q^i$ zu $d = q^i + 1$ wird das dem Gleichungssystem zugrunde liegende Polynom also immer um einen weiteren, als Faktor enthaltenen Term x^{q^i} „komplizierter“ und die Lösung dadurch vielleicht schwieriger.
Diese Begründung passt zudem auch zu der weiteren Beobachtung, dass bei $q = 3$ auch von $d = 2 \cdot q^{i-1}$ zu $d = q^i$ (allerdings etwas kleinere) Sprünge in der Laufzeitentwicklung zu erkennen sind. Denn für diesen Wechsel kommt im linearen Anteil $\sum_{q^i \leq d} \alpha_i x^{q^i}$ des zugrunde liegenden Polynoms ein neuer Term x^{q^i} hinzu. Bei $q = 2$ ist diese zweite Sorte von Sprüngen dagegen nicht zu erkennen, da hier $d = 2 \cdot q^{i-1}$ und $d = q^i$ zusammenfällt.

In der folgenden Tabelle werden nun die Ergebnisse der Untersuchungen zur Abhängigkeit der Laufzeit von d für weitere Werte von q und n zusammengefasst. Um zu große Datenmengen zu vermeiden, werden die Laufzeiten hier entsprechend der obigen Überlegung bezüglich $\lceil \log_q(d) \rceil$ angegeben. Zudem sind in der Zeile „zufällig“ als Vergleichswerte nochmal die Laufzeiten angegeben, die benötigt wurden, um ein zufällig erzeugtes, quadratisches Gleichungssystem mit denselben Parametern q und n zu lösen:

	$q=2$				$q=3$			$q=5$
$\lceil \log_q(d) \rceil$	$n=10$	$n=12$	$n=14$	$n=16$	$n=8$	$n=10$	$n=12$	$n=8$
1	0,003	0,003	0,003	0,003	0,025	0,115	0,527	0,841
2	0,024	0,066	0,182	0,499	0,122	0,983	12,127	1,948
3	0,043	0,147	0,494	1,193	0,267	3,203	28,520	3,013
4	0,073	0,357	1,848	6,050	0,273	5,330	83,722	3,016
5	0,103	0,780	5,167	20,659	0,272	5,344	132,056	3,024
6	0,103	0,783	8,888	41,079	0,274	5,354	132,599	
7	0,103	0,785	8,722	54,289	0,272	5,359	132,682	
8	0,103	0,792	8,708	55,177				
zufällig	0,102	0,789	8,705	57,307	0,274	5,205	132,671	3,018

An diesen Werten fällt vor allem auf, dass für kleine Werte von $\lceil \log_q(d) \rceil$ die Laufzeiten noch stark ansteigen bis zu einem Schwellwert, ab dem die gemessenen Laufzeiten alle der Laufzeit entsprechen, die für ein zufälliges System mit den gleichen Parametern q und n benötigt wird.
Daran ist besonders interessant, dass ein zufälliges System nach der Bemerkung in Abschnitt 2.3.1 immer als HFE-System zu einer Funktion φ aufgefasst werden kann, wobei für deren Grad $d = grad(\varphi)$ fast immer $\lceil \log_q(d) \rceil = n$ gilt. In Bezug auf die hier betrachteten Laufzeiten unterscheiden sich HFE-Systeme aber schon für deutlich kleinere d

(ungefähr für $\lceil \log_q(d) \rceil \geq \frac{n}{q}$) nicht mehr signifikant von zufälligen, quadratischen Gleichungssystemen.

Zuletzt bleibt noch zu untersuchen, wie sehr die Laufzeiten von der Größe q der verwendeten Grundkörper abhängen, da für größere q der in Abschnitt 3.4.1 beschriebene Vorteil der Gradbeschränkung ja immer mehr verschwindet.
Um zu vermeiden, dass diese Untersuchung durch die (für verschiedene q) unterschiedlichen Abhängigkeiten von $\lceil \log_q(d) \rceil$ gestört wird, wurden dazu nicht HFE-Systeme, sondern zufällig gewählte, quadratische Gleichungssysteme mit festem Parameter n untersucht. Für $n \in \{7, 8\}$ und $q \in \{2, 3, 5, 7, 11, 13\}$ wurden folgende Durchschnittszeiten gemessen:

q	2	3	5	7	11	13
$n = 7$	0,009	0,078	0,352	0,734	1,335	2,218
$n = 8$	0,018	0,273	3,016	4,890	8,966	13,345

Diese Werte zeigen, dass die Laufzeiten wie vermutet mit wachsendem q stark ansteigen. Allerdings scheint die Abhängigkeit nicht exponentiell zu sein.

Insgesamt können die Ergebnisse folgendermaßen zusammengefasst werden:
Die Komplexität der hier untersuchten Algorithmen ist so groß, dass von ihnen für HFE-Systeme mit Parametern q und n, die so gewählt sind, dass eine vollständige Suche unmöglich ist (also etwa $q^n \geq 2^{80}$), keine Gefahr ausgeht. Allerdings hat sich gezeigt, dass HFE-Systeme, die aus einem HFE-Polynom mit sehr kleinem Grad d entstehen, deutlich leichter zu lösen sind. Unter Umständen kann diese Schwachstelle von HFE-Systemen mit kleinem d durch gezielte Attacken ausgenutzt werden (vgl. Kapitel 5).

Kapitel 4

MQ unter speziellen Wahlen der Parameter

Im vorherigen Kapitel wurde die Lösung allgemeiner, polynomialer Gleichungssysteme betrachtet und auf den einfachen HFE-Fall mit genauso vielen Gleichungen wie Variablen angewendet. In diesem Kapitel wird nun das Lösen von Systemen beschrieben, die über- oder unterbestimmt sind. Solche Systeme kommen bei HFE beispielsweise bei den in Abschnitt 2.4 vorgestellten Perturbationen vor. Noch wichtiger sind manche der hier vorgestellten Algorithmen allerdings für die speziellen Attacken, die in Kapitel 5 beschrieben werden.

In diesem Kapitel sei für einen beliebigen, endlichen Körper $K = \mathbb{F}_q$ ein allgemeines quadratisches Gleichungssystem mit m Gleichungen in n Variablen gegeben, also ein System der Form

$$\begin{array}{lll} \sum\limits_{1\leq i\leq j\leq n} \beta_{ij1}x_ix_j & + \sum\limits_{1\leq i\leq n} \alpha_{i1}x_i & +\delta_1 = 0 \\ \vdots & \vdots & \vdots \\ \sum\limits_{1\leq i\leq j\leq n} \beta_{ijm}x_ix_j & + \sum\limits_{1\leq i\leq n} \alpha_{im}x_i & +\delta_m = 0 \end{array} \tag{4.1}$$

mit β_{ijk}, α_{ik}, $\delta_k \in K$. Gesucht ist dann nach mindestens einer Lösung $(a_1, \ldots, a_n) \in K^n$ im Grundkörper.

In Kapitel 2.5.2 wurde gezeigt, dass das zugrunde liegende, allgemeine Problem MQ, bei dem m und n beliebig gewählt werden dürfen, $\mathcal{NP}$-hart ist. In den folgenden Abschnitten werden Algorithmen vorgestellt, die dieses Problem für spezielle Wahlen von m und n in polynomieller oder zumindest subexponentieller Zeit lösen.

In Abschnitt 4.1 wird zunächst auf unterbestimmte Gleichungssysteme eingegangen, also Systeme mit $n > m$. In diesem Abschnitt wird ein Algorithmus vorgestellt, mit dem sich besonders stark unterbestimmte Systeme, genauer gesagt Systeme mit $n \geq m^2 + m$, zumindest über Körpern der Charakteristik 2 effizient lösen lassen.

Im darauffolgenden Abschnitt 4.2 werden dann verschiedene Techniken für $m > n$, also für überbestimmte Gleichungssysteme vorgestellt: die allgemein bekannte Linearisierung, mit der besonders stark überbestimmte Systeme gelöst werden können, und ihre Verallgemeinerungen, die Relinearisierung und der Algorithmus XL. Die beiden letzten Algorithmen bauen auf der Linearisierung auf, liefern aber auch für weniger stark überbestimmte Systeme oft noch schnelle Lösungsmöglichkeiten.
Die Analyse von XL gibt schließlich noch die Idee zum Algorithmus FXL, mit dem es vielleicht sogar möglich ist, auch Systeme mit $m = n$ in subexponentieller Zeit zu lösen. Dies wird in Abschnitt 4.2.4 kurz beschrieben.
Im abschließenden Abschnitt 4.3 wird dann noch erläutert, welche Auswirkungen die hier vorgestellten Algorithmen auf die Parameterwahl bei HFE, insbesondere bei Anwendung der Perturbationen haben.

4.1 Unterbestimmte Gleichungssysteme

In diesem Abschnitt werden quadratische Gleichungssysteme der Form (4.1) mit $n \geq m^2 + m$ betrachtet. Der im Folgenden vorgestellte Algorithmus zur Lösung solcher Systeme wurde von Goubin, Kipnis und Patarin in [GKP99] vorgestellt. Er arbeitet zwar über beliebigen Körpern K, ist aber nur für Körper der Charakteristik 2 wirklich effizient. Über Körpern mit Charakteristik $p > 2$ liefert der Algorithmus immerhin 2^m Werte, unter denen sich mindestens eine Lösung befindet.
In [GKP99] wurden allerdings einige Probleme, die bei der Anwendung dieses Algorithmus auftreten können, nur sehr kurz bzw. gar nicht erwähnt. Diese Probleme werden hier ausführlicher erläutert.

Die Idee des Algorithmus besteht darin, die große Anzahl an Variablen auszunutzen, um die quadratischen Polynome in (4.1) durch eine Variablentransformation

$$(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \cdot A \quad \text{mit } A \in K^{n \times n}$$

auf eine Pseudonormalform zu bringen. Durch die Gestalt dieser Pseudonormalform kann dann leichter eine Lösung für dieses neue System in den y_i gefunden werden und eine Rücktransformation dieser Lösung liefert dann natürlich eine Lösung für die x_i.

Die in $A = (a_{ij})_{1 \leq i,j \leq n}$ vorkommenden Koeffizienten werden nach und nach zeilenweise gewählt. Das Ziel dabei ist, A letztendlich so zu wählen, dass in dem transformierten System keine Produkte zwischen zwei verschiedenen der ersten m Variablen vorkommen, d.h. Terme der Form $y_i y_j$ mit $i, j \leq m$ und $i \neq j$ sollen eliminiert werden.
Dazu müssen Bedingungen für die im transformierten System vorkommenden Koeffizienten aufgestellt werden. Der Koeffizient bei $y_i y_j$ im k-ten Polynom des transformierten

Systems ist gerade

$$\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{i\mu} \cdot a_{j\nu}. \tag{4.2}$$

Die erste Zeile $(a_{11}, \ldots, a_{1n})$ von A kann beliebig von 0 verschieden gewählt werden. Die zweite Zeile wird dann (abhängig von der Wahl der ersten Zeile) so gewählt, dass die Produkte $y_1 y_2$ verschwinden. Mit (4.2) müssen also folgende Gleichungen erfüllt sein:

$$\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{1\mu} \cdot a_{2\nu} = 0 \qquad \text{für } k = 1, \ldots, m.$$

Dies sind m lineare Gleichungen in den n Variablen $a_{21}, \ldots, a_{2n}$, die somit wegen $n \geq m^2 + m$ auf jeden Fall eine nichttriviale Lösung liefern.

Im nächsten Schritt wird dann die dritte Zeile von A so gewählt, dass $y_1 y_3$ und $y_2 y_3$ verschwinden. Es muss also eine Lösung des Systems

$$\left.\begin{array}{l} \displaystyle\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{1\mu} \cdot a_{3\nu} = 0 \\ \displaystyle\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{2\mu} \cdot a_{3\nu} = 0 \end{array}\right\} \qquad \text{für } k = 1, \ldots, m$$

mit $2m$ Gleichungen in n Unbekannten gefunden werden, was wegen $n \geq m^2 + m$ natürlich ebenfalls auf jeden Fall möglich ist.

Auf diese Weise werden nun auch die folgenden Zeilen von A bestimmt: Im l-ten Schritt wird eine Lösung des Systems

$$\left.\begin{array}{l} \displaystyle\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{1\mu} \cdot a_{l\nu} = 0 \\ \qquad\vdots \qquad\qquad\qquad \vdots \\ \displaystyle\sum_{1\leq\mu\leq\nu\leq n} \beta_{\mu\nu k} \cdot a_{l-1,\mu} \cdot a_{l\nu} = 0 \end{array}\right\} \qquad \text{für } k = 1, \ldots, m$$

mit $(l-1)\cdot m$ Gleichungen in den n Variablen $a_{l1}, \ldots, a_{ln}$ in Abhängigkeit von den vorher gewählten, ersten $(l-1)$ Zeilen gesucht. So ergeben sich sukzessive für $l = 1, \ldots, m$ die ersten m Zeilen von A.

Zu beachten ist dabei noch, dass A eine Variablentransformation darstellen soll, also invertierbar gewählt werden muss. Dazu wird in [GKP99] gesagt, dass die wie oben beschrieben gewählten m Zeilen für ein zufällig gewähltes System (4.1) wahrscheinlich linear unabhängig sind.

Es ist sogar *immer* möglich, jede neue Zeile so zu wählen, dass das gesamte System linear unabhängig bleibt, wie folgendermaßen zu sehen ist: Wähle nach dem l-ten Schritt $(l = 1, \ldots, m)$ ein j_l mit $a_{l,j_l} \neq 0$ und setze die (eigentlich erst in späteren Schritten zu

bestimmenden) Variablen $a_{l+1,j_l}, \ldots, a_{m,j_l}$ auf 0. Dann bleibt für die Gleichungssysteme in den folgenden Schritten jeweils eine Variable weniger, im l-ten Schritt also ein System mit $(l-1)\cdot m$ Gleichungen in $n-l+1$ Variablen. Für $l \leq m$ ist aber

$$n-l+1 \underset{l\leq m}{\geq} n-m+1 \underset{n\geq m^2+m}{\geq} m^2+1 > (m-1)\cdot m \underset{m\geq l}{\geq} (l-1)\cdot m,$$

d.h. die übrigbleibenden Gleichungssysteme besitzen immer noch in jedem Fall eine nichttriviale Lösung. Zudem sind die m Spalten $a_{.,j_1}, \ldots, a_{.,j_m}$ alle von der Form

$$a_{.,j_l} = \begin{pmatrix} a_{1,j_l} \\ \vdots \\ a_{l,j_l} \neq 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

und somit in jedem Fall linear unabhängig. Damit müssen aber auch mindestens m (d.h. alle) Zeilen linear unabhängig sein.

Die restlichen Zeilen von A können beliebig gewählt werden, wobei nur noch beachtet werden muss, dass A nicht singulär sein darf. Dies kann beispielsweise auf eine ähnliche Weise erreicht werden, wie gerade für die ersten m Zeilen beschrieben. Die Transformation $(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \cdot A$ verwandelt (4.1) dann in ein System

$$\begin{array}{ccccc} \sum_{i=1}^{m} \gamma_{i1} y_i^2 & + & \sum_{i=1}^{m} L_{i1}(y_{m+1}, \ldots, y_n) \cdot y_i & + & Q_1(y_{m+1,}, \ldots, y_n) = 0 \\ \vdots & & \vdots & & \vdots \\ \sum_{i=1}^{m} \gamma_{im} y_i^2 & + & \sum_{i=1}^{m} L_{im}(y_{m+1}, \ldots, y_n) \cdot y_i & + & Q_m(y_{m+1,}, \ldots, y_n) = 0\ , \end{array}$$

wobei die L_{ij} bzw. Q_i Polynome aus $K[y_{m+1}, \ldots, y_n]$ vom Grad 1 bzw. 2 darstellen.

Bedingungen der Form $L_{ij} = 0$ für alle $i, j \in \{1, \ldots, m\}$ beschreiben für die Variablen $y_{m+1}, \ldots, y_n$ ein lineares Gleichungssystem mit m^2 Gleichungen in $n-m \geq m^2$ Variablen. Ist es möglich, eine Lösung $y_{m+1} = b_{m+1}, \ldots, y_n = b_n$ dieses Systems zu finden, so vereinfacht diese das obige System zu

$$\begin{array}{ll} \sum_{i=1}^{m} \gamma_{i1} y_i^2 & = -Q_1(b_{m+1,}, \ldots, b_n) \\ \vdots & \vdots \\ \sum_{i=1}^{m} \gamma_{im} y_i^2 & = -Q_m(b_{m+1,}, \ldots, b_n)\ . \end{array} \tag{4.3}$$

Der entscheidende Punkt (der leider in [GKP99] nicht erwähnt wird) ist, dass sowohl das System der $L_{ij} = 0$ als auch das neue System (4.3) — als lineares Gleichungssystem in den Variablen $z_i := {y_i}^2$ aufgefasst — nicht unbedingt lösbar sein muss. Genauer gesagt sind die Systeme genau dann lösbar ist, wenn der Rang ihrer Koeffizientenmatrix gleich dem Rang der erweiterten Koeffizientenmatrix ist. Etwa für das System (4.3) bedeutet dies, dass

$$Rang\,(\Gamma) = Rang\,(\,\Gamma \mid\, -Q\,) \quad \text{mit } \Gamma = \left(\gamma_{ij}\right)_{1\le i,j\le m} = \left(\sum_{1\le\mu\le\nu\le n} \beta_{\mu\nu j}\cdot a_{i\mu}a_{i\nu}\right)_{1\le i,j\le m}$$
$$\text{und } Q = \left(Q_1\left(b_{m+1,}\ldots,b_n\right),\ldots,Q_m\left(b_{m+1,}\ldots,b_n\right)\right)^t$$

gilt, was beispielsweise für eine invertierbare Matrix Γ erfüllt ist.

Folgendes Lemma gibt eine Abschätzung dafür, dass eine zufällig gewählte Matrix vollen Rang hat:

Lemma 4.1.1
Sei $n_1 \le n_2$ und $M \in K^{n_1\times n_2}$ (bzw. $M \in K^{n_2\times n_1}$) zufällig und gleichverteilt gewählt. Dann gilt

$$\Pr\left(M \textit{ hat den maximalen Rang } n_1\right) = \prod_{i=1}^{n_1}\left(1-\frac{1}{q^{n_2-i+1}}\right) \ge 1-\frac{n_1}{q}.$$

Beweis:
Sei $M = (m_{ij})_{\substack{1\le i\le n_1\\ 1\le j\le n_2}}$ und $M_i := (m_{i1},\ldots,m_{in_2})$ bezeichne die i-te Zeile von M. Ferner sei B_i das Ereignis

$$M_i \notin \langle M_1,\ldots,M_{i-1}\rangle_K \subseteq K^{n_2}.$$

Treten die Ereignisse $B_1,\ldots,B_{i-1}$ ein, ist $\dim_K\left(\langle M_1,\ldots,M_{i-1}\rangle_K\right) = i-1$ und damit $\left|\langle M_1,\ldots,M_{i-1}\rangle_K\right| = q^{i-1}$. Da $M_i \in K^{n_2}$ zufällig und gleichverteilt gewählt ist, folgt damit für alle i

$$\Pr\left(B_i \mid B_1,\ldots,B_{i-1}\right) = \frac{q^{n_2}-q^{i-1}}{q^{n_2}} = 1-\frac{1}{q^{n_2-i+1}}.$$

Damit gilt dann

$$\Pr\left(Rang\,(M)=n_1\right) = \Pr\left(B_1,\ldots,B_{n_1}\right) = \prod_{i=1}^{n_1}\Pr\left(B_i \mid B_1,\ldots,B_{i-1}\right)$$
$$= \prod_{i=1}^{n_1}\left(1-\frac{1}{q^{n_2-i+1}}\right) \ge \left(1-\frac{1}{q^{n_2-n_1+1}}\right)^{n_1} \ge 1-\frac{n_1}{q^{n_2-n_1+1}} \underset{n_2\ge n_1}{\ge} 1-\frac{n_1}{q}.$$

■

Wird davon ausgegangen, dass das ursprüngliche System zufällig gewählt war, und wurden die frei wählbaren Elemente der Transformationsmatrix A ebenfalls zufällig gewählt,

so kann annähernd davon ausgegangen werden, dass auch die Koeffizientenmatrizen der Systeme $L_{ij} = 0$ und (4.3) zufällig sind.
Nach Lemma 4.1.1 ist damit die Wahrscheinlichkeit, dass sie invertierbar und damit die Gleichungssysteme lösbar sind, für große Körper sehr groß. Auch für kleine Körper kann zumindest gezeigt werden, dass für $M \in K^{m \times m}$

$$\Pr\left(M \text{ invertierbar}\right) = \prod_{i=1}^{m} \left(1 - \frac{1}{q^i}\right) > \frac{1}{4}$$

gilt. Es kann also davon ausgegangen werden, dass es genügt, den Algorithmus einige Male durchzuführen, um an beiden problematischen Stellen lösbare Gleichungssysteme zu erhalten, so dass der Algorithmus bis hierhin schließlich mindestens eine Lösung für die ${y_i}^2$ liefert.

Ist $char\left(K\right) = 2$, dann ist $x \mapsto x^2$ ein Bijektion. Die Lösung für die y_i^2 liefert also direkt eine Lösung für die y_i und zusammen mit den oben bestimmten $y_{m+1} = b_{m+1}, \ldots, y_n = b_n$ somit eine Lösung des Gesamtsystems.

Ist dagegen $char\left(K\right) \neq 2$, so gibt es zu jedem y_i^2 bis zu zwei verschiedene Möglichkeiten für y_i. Aus jeder Lösung für die y_i^2 können also im Allgemeinen insgesamt bis zu 2^m verschiedene Belegungen für die $y_1, \ldots, y_n$ bestimmt werden, unter denen sich mindestens eine Lösung des ursprünglichen Systems befindet.

Die Laufzeit des hier beschriebenen Algorithmus wird im Wesentlichen durch das Lösen verschiedener linearer Gleichungssysteme mit höchstens m^2 Gleichungen und n Variablen und durch die nötige Anzahl der Wiederholungen bestimmt. Sie ist also nach den obigen Betrachtungen auf jeden Fall polynomiell in m und n.

4.2 Überbestimmte Gleichungssysteme

In diesem Abschnitt wird beschrieben, wie beim Lösen von quadratischen Gleichungssystemen ausgenutzt werden kann, dass ein System überbestimmt, also $m > n$ gegeben ist.

Die hier vorgestellten Techniken basieren in gewisser Weise alle auf einer Linearisierung, d.h. auf dem Ersetzen von Termen $x_{i_1} \cdot \ldots \cdot x_{i_d}$ vom Grad d durch neue Variablen $y_{i_1 \ldots i_d}$ vom Grad 1. Da meist eine quadratische Linearisierung durchgeführt wird und sich dabei die Anzahl der vorkommenden Variablen im Wesentlichen quadriert, ist es sinnvoll, als Maß der Überbestimmtheit des gegebenen Gleichungssystems

$$\epsilon := \frac{m}{n^2}$$

zu wählen.

Für den Fall $\epsilon > \frac{1}{2}$, also für besonders stark überbestimmte Systeme, genügt es im Allgemeinen, eine einfache **Linearisierung** durchzuführen. Diese Methode wird in Abschnitt 4.2.1 genauer vorgestellt.
Für kleinere ϵ dagegen sind ausgereiftere Techniken nötig, um die Systeme (zumindest in vielen Fällen) noch effizient zu lösen. Solche Techniken sind beispielsweise die **Relinearisierung**, die in Abschnitt 4.2.2 beschrieben wird, und der Algorithmus **XL**, der in Abschnitt 4.2.3 vorgestellt wird.
Anschließend wird in Abschnitt 4.2.4 noch der Algorithmus FXL vorgestellt, eine Variante von XL zum Lösen von schwach oder gar nicht überbestimmten Systemen, und in Abschnitt 4.2.5 werden abschließend XL und die Relinearisierung noch kurz verglichen.

Um die Beschreibungen der Algorithmen zu vereinfachen, wird angenommen, dass das ursprüngliche System keine linearen Terme in den x_i enthält, d.h. das zu lösende System ist von der Form

$$\begin{aligned} \sum_{1 \le i \le j \le n} \beta_{ij1} x_i x_j \quad &+ \delta_1 \;=\; 0 \\ \vdots \qquad\qquad\quad &\quad\; \vdots \\ \sum_{1 \le i \le j \le n} \beta_{ijm} x_i x_j \quad &+ \delta_m \;=\; 0. \end{aligned} \tag{4.4}$$

Die Algorithmen funktionieren allerdings in analoger Weise auch für beliebige quadratische Systeme.

Die Wahrscheinlichkeit, dass ein zufällig gewähltes, stark überbestimmtes System eine Lösung besitzt, ist gering. Da in der praktischen Anwendung aber ohnehin oft lösbare Gleichungssysteme betrachtet werden, wird in diesem Abschnitt vorausgesetzt, dass das betrachtete System eine Lösung besitzt.

4.2.1 Linearisierung

Die Idee der Linearisierung ist, das zu lösende, quadratische Gleichungssystem in ein leicht lösbares, lineares Gleichungssystem mit $\frac{n(n+1)}{2}$ Variablen zu transformieren, um anschließend aus den Lösungen des neuen Systems die Lösungen des ursprünglichen Systems zu erhalten.

Der Algorithmus verläuft nach folgendem Schema:

Algorithmus 4.2.1 (Linearisierung)
Eingabe: ein quadratisches Gleichungssystem $\mathcal{S}$ der Form (4.4)
Ausgabe: alle Lösungen $(b_1, \ldots, b_n)$ von $\mathcal{S}$

Linearisierung($\mathcal{S}$):

1. Ersetze in $\mathcal{S}$ alle vorkommenden Produkte $x_i x_j$ (mit $i \leq j$) durch neue Variablen y_{ij}; dies ergibt ein neues, nun lineares Gleichungssystem

$$\left.\begin{array}{rcl} \sum\limits_{1\leq i\leq j\leq n} \beta_{ij1} y_{ij} & + \delta_1 & = 0 \\ \vdots & & \vdots \\ \sum\limits_{1\leq i\leq j\leq n} \beta_{ijm} y_{ij} & + \delta_m & = 0 \end{array}\right\} \mathcal{S}'.$$

2. Bestimme die Lösungen von $\mathcal{S}'$ mit Hilfe einer Gauß-Elimination.

3. Für jede Lösung $(\ldots, a_{ij}, \ldots) \in K^{\frac{n(n+1)}{2}}$ von $\mathcal{S}'$, suche alle $(b_1, \ldots, b_n) \in K^n$ mit $a_{ij} = b_i b_j$ und überprüfe, ob diese eine Lösung von $\mathcal{S}$ darstellen.

Die **Korrektheit** dieses Algorithmus ist leicht nachzuvollziehen, da aufgrund der Konstruktion von $\mathcal{S}'$ zu jeder Lösung von $\mathcal{S}$ eine Lösung des neuen Systems $\mathcal{S}'$ existiert, die die Bedingungen $a_{ij} = b_i b_j$ erfüllt.

Die **Laufzeit** hängt neben der Gauß-Elimination in Schritt 2, die im Wesentlichen eine Zeit von $O\left(m^3\right)$ benötigt, vor allem von der Anzahl der Lösungen von $\mathcal{S}'$ ab, die in Schritt 3 durchgegangen werden müssen, und davon, wie effizient aus den $(\ldots, a_{ij}, \ldots)$ passende $(b_1, \ldots, b_n)$ bestimmt werden können.

Zunächst zur Bestimmung von $(b_1, \ldots, b_n)$:
Sei $(\ldots, a_{ij}, \ldots) \in K^{\frac{n(n+1)}{2}}$ eine der Lösungen von $\mathcal{S}'$. Existiert dazu eine Lösung $(b_1, \ldots, b_n) \in K^n$ von $\mathcal{S}$ mit $a_{ij} = b_i \cdot b_j$, so lässt sich diese leicht aus den a_{ij} bestimmen:

- Ist $char\,(K) = 2$, so existiert (vgl. Abschnitt 4.1) zu jedem a_{ii} genau ein b_i mit ${b_i}^2 = a_{ii}$. Für diese b_i kann dann leicht überprüft werden, ob sie das ursprüngliche System lösen.

- Ist $char\,(K) \neq 2$, so gibt es zu $(\ldots, a_{ij}, \ldots)$ höchstens zwei Wahlen für $(b_1, \ldots, b_n)$, die $a_{ij} = b_i \cdot b_j$ erfüllen. Diese lassen sich folgendermaßen bestimmen:
  Ist $a_{ii} = 0$ für alle i, so ist $(b_1, \ldots, b_n) = (0, \ldots, 0)$ die einzige Möglichkeit.
  Ansonsten sei $r := \min\{i \mid 1 \leq i \leq n\,,\, a_{ii} \neq 0\}$.
  Ist a_{rr} ein Quadrat in K, existieren genau zwei Werte $\widetilde{a}, -\widetilde{a} \in K$ mit ${\widetilde{a}}^2 = a_{rr} = (-\widetilde{a})^2$. In diesem Fall können nur $(b_1, \ldots, b_n)$ mit

$$b_i = \begin{cases} 0 & \text{für } i < r, \\ \pm\widetilde{a} & \text{für } i = r, \\ \frac{a_{ri}}{b_r} & \text{für } i > r \end{cases}$$

die Bedingungen $a_{ij} = b_i b_j$ erfüllen.
Ist dagegen a_{rr} kein Quadrat in K, so gibt es gar kein $(b_1, \ldots, b_n) \in K$ mit $a_{ij} = b_i \cdot b_j$, da $a_{rr} = b_r^2$ nicht erfüllbar ist.
Es gibt also zu jedem $(\ldots, a_{ij}, \ldots)$ höchstens zwei Tupel $(b_1, \ldots, b_n)$, die die Bedingungen $a_{ij} = b_i b_j$ erfüllen und sie können auf diese Weise effizient bestimmt werden. Anschließend kann dann leicht überprüft werden, ob diese Tupel zudem das ursprüngliche System lösen.

Die Effizienz einer Linearisierung hängt also im Wesentlichen noch davon ab, wieviele Lösungen $(\ldots, a_{ij}, \ldots)$ das System $\mathcal{S}'$ liefert. Unter der Annahme, dass $\mathcal{S}$ und damit auch $\mathcal{S}'$ lösbar ist, kann die Dimension l des Lösungsraumes von $\mathcal{S}'$ angegeben werden als

$$l = \frac{n\,(n+1)}{2} - m',$$

wobei m' den Rang der Koeffizientenmatrix

$$\begin{pmatrix} \beta_{111} & \beta_{121} & \cdots & \beta_{nn1} \\ \vdots & \vdots & & \vdots \\ \beta_{11m} & \beta_{12m} & \cdots & \beta_{nnm} \end{pmatrix}$$

bezeichnet.

Ist das ursprüngliche Gleichungssystem zufällig gewählt, so ist nach Lemma 4.1.1 für große Körper K

$$m' = \min\left(\frac{n\,(n+1)}{2}, m\right)$$

zu erwarten. Bei einer großen Überbestimmtheit von $\epsilon \geq \frac{1}{2} + \frac{1}{2n}$, ist damit wegen

$$m = \epsilon n^2 \geq \left(\frac{1}{2} + \frac{1}{2n}\right) n^2 = \frac{n^2 + n}{2}$$

auch eine Lösungsdimension von $l = \frac{n(n+1)}{2} - m' = 0$ zu erwarten, d.h. eine eindeutige Lösung für $\mathcal{S}'$.
Ist K vergleichsweise klein, so ist nach Lemma 4.1.1 die Wahrscheinlichkeit für $m' = \min\left(\frac{n(n+1)}{2}, m\right)$ deutlich geringer, und damit nicht unbedingt $l = 0$ zu erwarten. Liegt allerdings m' „nahe bei“ $\min\left(\frac{n(n+1)}{2}, m\right)$ und ist damit l für $\epsilon \geq \frac{1}{2} + \frac{1}{2n}$ sehr klein, so ist für einen kleinen Körper K die Gesamtzahl q^l der Lösungen immer noch ziemlich klein. Unter der Annahme $m' \approx \min\left(\frac{n(n+1)}{2}, m\right)$ können also trotzdem im Algorithmus 4.2.1 alle Lösungen von $\mathcal{S}'$ durchlaufen werden. Ob diese Annahme allerdings sinnvoll ist, wird hier nicht näher betrachtet.

Für den Fall $\epsilon \geq \frac{1}{2} + \frac{1}{2n}$ hängt die erwartete Laufzeit der Linearisierung somit vor allem von der für die Lösung von $\mathcal{S}'$ benötigten Zeit ab. Diese beträgt wie oben erwähnt im Wesentlichen $O\left(m^3\right)$.

Ist die Überbestimmtheit aber geringer, etwa $\epsilon < \frac{1}{2}$, hat das System $\mathcal{S}'$ q^l, also exponentiell viele, Lösungen mit

$$l = \underbrace{\frac{n\,(n+1)}{2}}_{>\frac{1}{2}n^2} - \underbrace{m'}_{\leq m=\epsilon n^2} > \left(\frac{1}{2} - \epsilon\right) n^2.$$

Viele der Lösungen von $\mathcal{S}'$ führen nicht zu Lösungen von $\mathcal{S}$ und es ist zu ineffizient, diese alle durchzuprobieren, um eine gültige Lösung für $\mathcal{S}$ zu finden.
Deshalb müssen für solche Gleichungssysteme mit $\epsilon < \frac{1}{2}$ andere Techniken angewendet werden, wie beispielsweise die Relinearisierung, die im folgenden Abschnitt beschrieben wird.

Bemerkung 4.2.2
Eine Linearisierung kann auch auf polynomiale Gleichungssysteme höheren Grades angewendet werden. Dabei werden dann die Monome $x_{i_1} \cdot \ldots \cdot x_{i_d}$ durch neue Variablen $y_{i_1 \ldots i_d}$ ersetzt. Bei einem Gleichungssystem mit Polynomen vom Grad d, steigt dadurch allerdings die Anzahl der neuen Variablen im Allgemeinen auf $\binom{n+d-1}{d} \approx \frac{n^d}{d!}$. Es werden also auch deutlich mehr Gleichungen benötigt, um mit guter Wahrscheinlichkeit eine Lösung in effizienter Zeit zu bekommen.

4.2.2 Relinearisierung

Die von Kipnis und Shamir in [KS99] vorgestellte Relinearisierung stellt eine Erweiterung der Linearisierungstechnik dar. Sie versucht, aus den unter Umständen größtenteils überflüssigen Zwischenlösungen, die bei der oben beschriebenen Linearisierung entstehen, diejenigen herauszufinden, die wirklich zu Lösungen des ursprünglichen Systems führen. Die in diesem Abschnitt durchgeführte Analyse der Relinearisierung basiert auf der Analyse in [CKPS00], ist aber in weiten Teilen ausführlicher.

Grundlage für die Relinearisierung ist die Beobachtung, dass die bei der Linearisierung neu eingeführten y_{ij} nicht algebraisch unabhängig sind, sondern durch ihre Darstellung als $y_{ij} = x_i x_j$ stark voneinander abhängen. Genauer gesagt gilt die folgende Beziehung:

Seien $d \in \{4, 6, 8, \ldots\}$ und $(i_1, \ldots, i_d), (j_1, \ldots, j_d) \in \{1, \ldots, n\}^d$ zwei d-Tupel mit $(i_1, \ldots, i_d) \sim (j_1, \ldots, j_d)$, d.h. es existiere ein $\sigma \in S_d$, so dass $i_k = j_{\sigma(k)}$ für $k = 1, \ldots, d$ gilt. Dann gilt folgende Gleichung:

$$y_{i_1 i_2} \cdot \ldots \cdot y_{i_{d-1} i_d} = x_{i_1} \cdot x_{i_2} \cdot \ldots \cdot x_{i_d} = x_{j_1} \cdot x_{j_2} \cdot \ldots \cdot x_{j_d} = y_{j_1 j_2} \cdot \ldots \cdot y_{j_{d-1} j_d}.$$

Allerdings können solche Gleichungen nur für Tupel $(i_1, \ldots, i_d), (j_1, \ldots, j_d)$ mit $i_{2k-1} \leq i_{2k}$ und $j_{2k-1} \leq j_{2k}$ für $k = 1, \ldots, \frac{d}{2}$ gebildet werden, da die Variablen y_{ij} nur für $x_i x_j$ mit $i \leq j$ eingeführt wurden.

Zudem kann ein und diesselbe solche Gleichung natürlich durch Vertauschen einzelner Faktoren y_{ij} oder durch Vertauschen der beiden Seiten mit Hilfe verschiedener d-Tupel-Paare erzeugt werden. Um zu vermeiden, dass solche Gleichungen doppelt betrachtet werden, wird für die Gleichungen, die bei einer Relinearisierung verwendet werden, zudem noch gefordert, dass die beiden Mengen von Indexpaaren $(i_1, i_2), (i_3, i_4), \ldots, (i_{d-1}, i_d)$ und $(j_1, j_2), \ldots, (j_{d-1}, j_d)$ bezüglich der lexikographischen Ordnung auf den Paaren aufsteigend sortiert sind (um Vertauschungen der y_{ij} zu vermeiden), und dass $(i_1, \ldots, i_d)$ als Gesamtes auch lexikographisch kleiner ist als $(j_1, \ldots, j_d)$ (um Vertauschungen der Seiten zu vermeiden).

Dies führt zu folgender Definition:

Definition 4.2.3
Sei $d \in \{4, 6, 8, \ldots\}$.

- *Ein* d*-Tupel* $(i_1, \ldots, i_d) \in \{1, \ldots, n\}^d$ *heißt* ***gültig****, falls gilt:*

  i) $i_{2k-1} \leq i_{2k}$ *für alle* $k = 1, \ldots, \frac{d}{2}$ *(nur gültige Variablen* $y_{i_{2k-1}i_{2k}}$*),*

  ii) $(i_{2k-1} < i_{2l-1})$ *oder* $(i_{2k-1} = i_{2l-1}$ *und* $i_{2k} \leq i_{2l})$ *für alle* $1 \leq k \leq l \leq \frac{d}{2}$ *(lexikographische Anordnung der Indexpaare).*

- *Seien* $(i_1, \ldots, i_d) \sim (j_1, \ldots, j_d) \in \{1, \ldots, n\}^d$ *zwei gültige* d*-Tupel, die durch eine Permutation auseinander hervorgehen und ferner sei* $(i_1, \ldots, i_d)$ *lexikographisch kleiner als* $(j_1, \ldots, j_d)$*. Dann heißt*

  $$y_{i_1 i_2} \cdot \ldots \cdot y_{i_{d-1} i_d} = y_{j_1 j_2} \cdot \ldots \cdot y_{j_{d-1} j_d}$$

  eine ***gültige Gleichung*** *vom Grad* d*. Für* $I = (i_1, \ldots, i_d)$ *und* $J = (j_1, \ldots, j_d)$ *wird diese Gleichung auch kurz mit* $y_I = y_J$ *bezeichnet.*

Solche gültigen Gleichungen können dann zu einem neuen Gleichungssystem $\widetilde{\mathcal{S}}$ zusammengefasst werden. $\widetilde{\mathcal{S}}$ beschreibt also zusätzliche Bedingungen, die die Lösungen für y_{ij} in $\mathcal{S}'$, die zu Lösungen für die x_i in $\mathcal{S}$ gehören, von denen unterscheiden, die nicht zu Lösungen von $\mathcal{S}$ gehören.
Dieses System ist dann wieder ein polynomiales Gleichungssystem (für $d = 4$ wieder quadratisch) und somit kann versucht werden, es durch eine Linearisierung oder eine erneute Relinearisierung zu lösen.

Dies führt insgesamt zu folgendem Algorithmus:

Algorithmus 4.2.4 (Relinearisierung)
Eingabe: ein quadratisches Gleichungssystem $\mathcal{S}$ der Form (4.4), $d \in \{4, 6, 8, \ldots\}$
Ausgabe: alle Lösungen $(b_1, \ldots, b_n)$ von $\mathcal{S}$

Relinearisierung$(\mathcal{S}, d)$:

1. Ersetze in $\mathcal{S}$ alle vorkommenden Produkte $x_i x_j$ (mit $i \leq j$) durch neue Variablen y_{ij}; dies ergibt ein neues, nun lineares Gleichungssystem

$$\left.\begin{array}{rcl} \sum_{1\leq i\leq j\leq n} \beta_{ij1} y_{ij} & + \delta_1 & = 0 \\ \vdots & & \vdots \\ \sum_{1\leq i\leq j\leq n} \beta_{ijm} y_{ij} & + \delta_m & = 0 \end{array}\right\} \mathcal{S}'.$$

2. Bestimme die Lösungen von $\mathcal{S}'$ mit Hilfe einer Gauß-Elimination und beschreibe diese durch Parameter $z_1, \ldots, z_l$.

3. Betrachte gültige Gleichungen der Form

$$y_{i_1 i_2} \cdot \ldots \cdot y_{i_{d-1} i_d} = y_{j_1 j_2} \cdot \ldots \cdot y_{j_{d-1} j_d}.$$

   In diese Gleichungen setze dann die durch $z_1, \ldots, z_l$ parametrisierten Lösungen für die y_{ij} ein und erhalte ein polynomiales Gleichungssystem $\widetilde{\mathcal{S}}$ vom Grad $\frac{d}{2}$.

4. a) Falls $\widetilde{\mathcal{S}}$ dazu stark genug überbestimmt ist (etwa $\widetilde{\epsilon} > \frac{1}{2}$), löse nun $\widetilde{\mathcal{S}}$ mit Hilfe einer Linearisierung und resubstituiere die so erhaltenen Lösungen, um daraus die Lösungen für das ursprüngliche System zu erhalten.
   oder
   b) Ist $\widetilde{\mathcal{S}}$ nicht stark genug überbestimmt, löse $\widetilde{\mathcal{S}}$ mit einer erneuten Relinearisierung, d.h. starte erneut bei Schritt 1 mit $\widetilde{\mathcal{S}}$ anstelle von $\mathcal{S}$.

Bemerkung 4.2.5
Ist kein festes d vorgegeben, so besteht in Schritt 4 b) auch die Möglichkeit statt rekursiv mit $\widetilde{\mathcal{S}}$ von vorne zu beginnen, zunächst zu Schritt 3 zurückzugehen und den Grad d der betrachteten Polynome zu erhöhen, so dass sich mehr Gleichungen ergeben.

Der Vorteil an diesem Aufbau des Relinearisierungs-Algorithmus ist, dass die Schritte 1 und 2 genau mit den ersten beiden Schritten einer gewöhnlichen Linearisierung übereinstimmen. Es kann also zunächst immer versucht werden, eine Linearisierung durchzuführen. Stellt sich die dabei erhaltene Anzahl an Lösungen als zu groß heraus, d.h. ist die Lösungsdimension l in Schritt 2 zu groß, so kann direkt mit Schritt 3 der Relinearisierung fortgefahren werden.

Die Lösung des ursprünglichen, polynomialen Gleichungssystems $\mathcal{S}$ mit m Gleichungen in n Variablen wird bei einer Relinearisierung zurückgeführt auf die Lösung eines anderen

polynomialen Gleichungssystems $\widetilde{\mathcal{S}}$ in $\widetilde{m}$ Gleichungen mit $\widetilde{n} := l$ Variablen. Die Hoffnung bei diesem Vorgehen ist, dass das neue System effizienter lösbar als das ursprüngliche System ist.
Der entscheidende Punkt für die Lösbarkeit des neuen Systems $\widetilde{\mathcal{S}}$ und damit für die Effizienz der Relinearisierung ist seine Überbestimmtheit oder genauer — wie in Abschnitt 4.2.1 gesehen — der Rang $\widetilde{m}'$ der zugehörigen Koeffizientenmatrix. Dieser Rang ist beschränkt durch die maximale Anzahl linear unabhängiger, gültiger Gleichungen. Dafür gilt folgendes Lemma:

Lemma 4.2.6
Sei $\nu(i_1, \ldots, i_d)$ die Anzahl der gültigen d-Tupel $(j_1, \ldots, j_d)$ mit $(i_1, \ldots, i_d) \sim (j_1, \ldots, j_d)$. Dann gilt:

i) Die Anzahl der gültigen Gleichungen vom Grad d beträgt $\sum_{1 \le i_1 \le \ldots \le i_d \le n} \binom{\nu(i_1,\ldots,i_d)}{2}$.

ii) Unter diesen Gleichungen gibt es höchstens $\sum_{1 \le i_1 \le \ldots \le i_d \le n} (\nu(i_1, \ldots, i_d) - 1)$ linear unabhängige Gleichungen.

Beweis:
Die Menge $\{1, \ldots, n\}^d$ aller d-Tupel zerfällt bezüglich der Äquivalenzrelation $\sim$ in Äquivalenzklassen, die durch das Repräsentantensystem

$$\left\{(i_1, \ldots, i_d) \in \{1, \ldots, n\}^d \mid i_1 \le \ldots \le i_d\right\}$$

dargestellt werden können. In jeder Äquivalenzklasse gibt es $\nu(i_1, \ldots, i_d)$ gültige d-Tupel und genau aus jedem ungeordneten Paar davon kann eine gültige Gleichung erzeugt werden. Dies ergibt Behauptung i).
Weiter ist $\nu(i_1, \ldots, i_d) \ge 1$, da $I := (i_1, \ldots, i_d)$ mit $i_1 \le \ldots \le i_d$ immer gültig ist. Jede gültige Gleichung $y_J = y_{J'}$ zu zwei d-Tupeln $J = (j_1, \ldots, j_d)$ und $J' = (j'_1, \ldots, j'_d)$ mit $J \sim I \sim J'$ kann aber immer aus den zwei Gleichungen vom Typ $y_I = y_J$ und $y_I = y_{J'}$ linear kombiniert werden. Es genügt also die $\nu(I) - 1$ Gleichungen vom Typ $y_I = y_J$ mit $J \sim I$, $J \neq I$ zu betrachten. Daraus folgt Behauptung ii). ■

Lemma 4.2.6 ii) liefert also eine obere Schranke für $\widetilde{m}'$. Im Folgenden wird $\widetilde{m}'$ nun für spezielle Wahlen von d näher bestimmt und untersucht, was dies für die Lösbarkeit von polynomialen Gleichungssystemen mittels einer Relinearisierung bedeutet.

Grad 4-Relinearisierung

Die Analyse einer Grad 4-Relinearisierung, also einer Relinearisierung mit gültigen Gleichungen vom Grad $d = 4$, ist aufgrund der einfachen Struktur der gültigen Gleichungen noch recht übersichtlich. Dazu werden zunächst die gültigen 4-Tupel abgezählt:

Lemma 4.2.7

Für $(i_1, i_2, i_3, i_4) \in \{1, \ldots, n\}^4$ *mit* $i_1 \leq i_2 \leq i_3 \leq i_4$ *ist*

$$\nu(i_1, i_2, i_3, i_4) = \begin{cases} 3, & \textit{falls } i_1, i_2, i_3, i_4 \textit{ paarweise verschieden sind,} \\ 2, & \textit{falls mindestens ein Wert genau zweimal vorkommt,} \\ 1, & \textit{falls ein Wert mindestens dreimal vorkommt.} \end{cases}$$

Beweis:

Ein 4-Tupel $(j_1, j_2, j_3, j_4) \sim (i_1, i_2, i_3, i_4)$ ist genau dann gültig, wenn $j_1 \leq j_2$, $j_3 \leq j_4$ und ($j_1 < j_3$ oder ($j_1 = j_3$ und $j_2 \leq j_4$)) gilt. Damit lässt sich leicht folgende Tabelle aufstellen, in der die verschiedenen möglichen Fälle für $i_1 \leq i_2 \leq i_3 \leq i_4$ durch $a, b, c, d \in \{1, \ldots, n\}$ mit $a < b < c < d$ dargestellt sind:

$(i_1, \ldots, i_4)$	gültige Permutationen von $(i_1, \ldots, i_4)$
(a, b, c, d)	$(a, b, c, d), (a, c, b, d), (a, d, b, c)$
(a, a, b, c)	$(a, a, b, c), (a, b, a, c)$
(a, b, b, c)	$(a, b, b, c), (a, c, b, b)$
(a, b, c, c)	$(a, b, c, c), (a, c, b, c)$
(a, a, b, b)	$(a, a, b, b), (a, b, a, b)$
(a, a, a, b)	(a, a, a, b)
(a, b, b, b)	(a, b, b, b)
(a, a, a, a)	(a, a, a, a)

■

Da es $\binom{n}{4}$ 4-Tupel des Typs (a, b, c, d), je $\binom{n}{3}$ 4-Tupel der Typen $(a, a, b, c), (a, b, b, c)$ und (a, b, c, c) und $\binom{n}{2}$ 4-Tupel vom Typ (a, a, b, b) gibt, folgt damit die Anzahl linear unabhängiger, gültiger Gleichungen vom Grad 4:

Satz 4.2.8

Es können genau

$$2 \cdot \binom{n}{4} + 3 \cdot \binom{n}{3} + \binom{n}{2} = \frac{1}{12}\left(n^4 - n^2\right)$$

linear unabhängige, gültige Gleichungen vom Grad 4 *gefunden werden.*

Beweis:

Die Abschätzung $\leq 2 \cdot \binom{n}{4} + 3 \cdot \binom{n}{3} + \binom{n}{2}$ folgt direkt aus Lemma 4.2.6 *ii*), Lemma 4.2.7 und der Bemerkung vor diesem Satz. Dass zudem die $2 \cdot \binom{n}{4} + 3 \cdot \binom{n}{3} + \binom{n}{2}$ Gleichungen in

$$\left\{ y_{i_1 i_2} y_{i_3 i_4} = y_{j_1 j_2} y_{j_3 j_4} \;\middle|\; \begin{array}{c} (i_1, i_2, i_3, i_4) \sim (j_1, j_2, j_3, j_4) \text{ gültig} \\ \text{und verschieden, } i_1 \leq i_2 \leq i_3 \leq i_4 \end{array} \right\}$$

alle linear unabhängig sind, ist leicht einzusehen, da es zu jedem Indextupel $(i_1, \ldots, i_4)$ nach Lemma 4.2.7 immer höchstens zwei Gleichungen in dieser Menge gibt. ■

Im dritten Schritt der Relinearisierung werden anschließend die y_{ij} durch ihre in $z_1, \ldots, z_l$ parametrisierten Lösungen ersetzt, so dass sich ein quadratisches Gleichungssystem $\widetilde{\mathcal{S}}$ mit $\widetilde{m} = \frac{1}{12}\left(n^4 - n^2\right)$ Gleichungen in den $\widetilde{n} := l$ Variablen $z_1, \ldots, z_{\widetilde{n}}$ ergibt. Zu beachten ist dabei, dass in diesem System die z_i nicht nur in Produkten $z_i z_j$ auftreten können, sondern auch einzeln. Die anschließende Linearisierung ergibt also ein lineares System mit $\frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}$ Variablen.
Durch das Ersetzen der y_{ij} durch Linearkombinationen der z_i und die damit verbundene Reduktion der Anzahl der Variablen, geht im Allgemeinen die lineare Unabhängigkeit der $\widetilde{m}$ Gleichungen verloren. In [CKPS00] wird allerdings aufgrund von Experimenten die Vermutung aufgestellt, dass es zumindest für große Körper K sehr wahrscheinlich ist, dass der Rang der Koeffizientenmatrix maximal, also $\widetilde{m}' = \min\left(\widetilde{m}, \frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}\right)$ ist.

Für die nun folgenden Abschätzungen der Effizienz einer Grad 4-Relinearisierung wird angenommen, dass beide linearen Gleichungssysteme den maximal möglichen Rang erreichen, dass also sowohl $m' = \min\left(m, \frac{n(n+1)}{2}\right)$ als auch $\widetilde{m}' = \min\left(\widetilde{m}, \frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}\right)$ gilt.

Damit schon eine einzige Runde einer Relinearisierung ein eindeutig lösbares, lineares Gleichungssystem liefert, muss $\widetilde{l} := \frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n} - \widetilde{m}' = 0$, unter der obigen Annahme also $\widetilde{m} \geq \frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}$ gelten. Die folgende Tabelle gibt für kleine Werte von n die minimalen Werte für m an, die dies erfüllen:

n	m	$\frac{n(n+1)}{2} - m' = l = \widetilde{n}$	$\frac{n^4-n^2}{12} = \widetilde{m}$	$\frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}$	$\epsilon = \frac{m}{n^2}$
6	8	13	105	104	$\approx 0,22$
8	12	24	336	324	$\approx 0,19$
10	16	39	825	819	$0,16$
15	30	90	4200	4185	$\approx 0,13$

Asymptotisch, d.h. für große n, ist $\widetilde{m} \approx \frac{n^4}{12}$ und $\widetilde{n} \approx \frac{n^2}{2} - m = n^2\left(\frac{1}{2} - \epsilon\right)$ mit $\epsilon = \frac{m}{n^2}$. In die Bedingung $\widetilde{m} \geq \frac{\widetilde{n}(\widetilde{n}+1)}{2} + \widetilde{n}$ eingesetzt, ergibt dies

$$\frac{n^4}{12} \geq \frac{n^4\left(\frac{1}{2} - \epsilon\right)^2 + 3 \cdot n^2\left(\frac{1}{2} - \epsilon\right)}{2}.$$

Für große n ist diese Ungleichung erfüllt, falls

$$\frac{1}{6} > \left(\frac{1}{2} - \epsilon\right)^2 \quad \text{d.h.} \quad \epsilon > \frac{1}{2} - \frac{1}{\sqrt{6}} \approx 0,1$$

gilt. Mit Hilfe eines einzelnen Schritts einer Grad 4-Relinearisierung können also asymptotisch gesehen schon Gleichungssysteme mit $\epsilon > 0,1$ effizient gelöst werden, während für die einfache Linearisierung aus Abschnitt 4.2.1 noch $\epsilon > \frac{1}{2}$ benötigt wurde.

Relinearisierungen höheren Grades

Auch für Relinearisierungen höheren Grades kann der Wert der Schranke aus Lemma 4.2.6 *ii*) exakt bestimmt werden. Allerdings ist der Aufwand aufgrund der vielen zu betrachtenden Typen von d-Tupeln deutlich größer. Schon für Grad $d = 6$ gibt es beispielsweise 32 verschiedene Typen von 6-Tupeln, die betrachtet werden müssen und zu denen jeweils 1 bis 15 verschiedene gültige Permutationen existieren. Ohne die genaue Rechnung durchzuführen wird deshalb hier nur der exakte Wert für die obere Schranke aus Lemma 4.2.6 *ii*) für den Fall $d = 6$ angegeben. In diesem Fall gilt

$$\widetilde{m}' \leq \frac{7}{360}n^6 + \frac{1}{24}n^5 + \frac{5}{72}n^4 - \frac{1}{24}n^3 - \frac{4}{45}n^2.$$

Analog zur Grad 4-Relinearisierung gibt diese Abschätzung Anlass zu der Hoffnung, dass asymptotisch betrachtet, ungefähr $\frac{7}{360}n^6$ linear unabhängige Gleichungen vom Grad 6 gefunden werden können. Aufgrund dieser Annahme behaupten Kipnis und Shamir in [KS99] sogar, dass mit einer Grad 6-Relinearisierung Gleichungssysteme mit $\epsilon \geq 0,008$ gelöst werden könnten.

Leider ist diese Annahme aber falsch, denn alle gültigen Gleichungen können immer linear aus sogenannten speziellen Gleichungen kombiniert werden:

Definition 4.2.9
Eine gültige Gleichung

$$y_{i_1 i_2} \cdot \ldots \cdot y_{i_{d-1} i_d} = y_{j_1 j_2} \cdot \ldots \cdot y_{j_{d-1} j_d}$$

heißt ***speziell****, wenn*

$$\left| \left\{ y_{i_{2k-1} i_{2k}} \mid k = 1, \ldots, \frac{d}{2} \right\} \cap \left\{ y_{j_{2k-1} j_{2k}} \mid k = 1, \ldots, \frac{d}{2} \right\} \right| = \frac{d}{2} - 2$$

gilt, die $y_{i_{2k-1} i_{2k}}$ auf der linken Seite mit den $y_{j_{2k-1} j_{2k}}$ auf der rechten Seite also bis auf genau zwei Variablen übereinstimmen.

Für solche speziellen Gleichungen zeigt Courtois in [CKPS00] folgenden Satz:

Satz 4.2.10
Jede gültige Gleichung kann aus speziellen Gleichungen desselben Grades linear kombiniert werden.

Der Beweis dieses Satzes beruht im Wesentlichen auf der Idee, dass sich beliebige Permutationen als Produkt von Transpositionen darstellen lassen und dass diese Darstellung auf gültige und spezielle Gleichungen übertragbar ist.
Dieser Satz liefert direkt folgende, deutlich verbesserte, obere Schranke für $\widetilde{m}'$:

Korollar 4.2.11

Für $d \in \{4, 6, 8, \ldots\}$ gilt

$$\widetilde{m}' \leq 2 \cdot \binom{\frac{d}{2}}{2} \cdot \binom{n+d-1}{d}.$$

Beweis:

Nach Satz 4.2.10 ist $\widetilde{m}'$ durch die Anzahl spezieller Gleichungen beschränkt, es genügt also, diese abzuzählen.

Wie im Beweis von Lemma 4.2.6 schon gezeigt, genügt es beim Abzählen von linear unabhängigen Gleichungen, diejenigen Gleichungen zu betrachten, deren linke Seite durch ein d-Tupel $(i_1, \ldots, i_d)$ mit $i_1 \leq \ldots \leq i_d$ gegeben ist. Insgesamt gibt es $\binom{n+d-1}{d}$ Möglichkeiten, ein d-Tupel $(i_1, \ldots, i_d)$ mit $1 \leq i_1 \leq \ldots \leq i_d \leq n$ zu wählen. Weiter gibt es jeweils höchstens $\binom{\frac{d}{2}}{2}$ Möglichkeiten aus den $\frac{d}{2}$ Paaren $(i_1, i_2), \ldots, (i_{d-1}, i_d)$ zwei auszuwählen, in denen sich die linke von der rechten Seite unterscheidet. Zwei solche Paare können schließlich aber maximal auf drei gültige Arten permutiert werden (vgl. Lemma 4.2.7). Insgesamt gibt es somit höchstens $2 \cdot \binom{\frac{d}{2}}{2} \cdot \binom{n+d-1}{d}$ spezielle Gleichungen. ■

Beispielsweise für $d = 6$ liefert dieses Korollar eine Schranke von

$$\widetilde{m}' \leq \frac{1}{120}n^6 + \frac{1}{8}n^5 + \frac{17}{24}n^4 + \frac{15}{8}n^3 + \frac{137}{60}n^2 + n.$$

Es kann also nur deutlich weniger (asymptotisch ungefähr $\frac{1}{21}$) linear unabhängige, gültige Gleichungen geben als Kipnis und Shamir ursprünglich angenommen haben. Eine Erhöhung des verwendeten Grades bringt somit nicht so große Vorteile wie zunächst erwartet.

Dennoch scheint es sich zumindest etwas zu lohnen, Gleichungen höheren Grades zu betrachten. In [CKPS00] befindet sich eine Tabelle, in der die Parameter-Kombinationen aufgeführt sind, für die experimentell gezeigt wurde, dass sie mit einer Grad 6-Relinearisierung gelöst werden können. Dort sind unter anderem folgende Werte angegeben:

n	4	5	6	7	8	9
m	5	6	7	9	10	11

Verglichen mit den Werten aus der Tabelle auf Seite 104 zeigt sich also, dass die Grad 6-Relinearisierung wirklich schwächer überbestimmte Gleichungssysteme (d.h. Systeme mit kleinerem ϵ) lösen kann als eine Grad 4-Relinearisierung.

Der Nachteil einer Relinearisierung mit größerem d ist natürlich immer, dass das letztendlich zu lösende, lineare System ebenfalls deutlich größer wird. Bei der Wahl des Grades ist also immer abzuwägen zwischen einer möglichen besseren Lösbarkeit durch einen höheren Grad und einem kleineren, resultierenden System und damit einer besseren Effizienz bei einem kleineren Grad.

4.2.3 XL

Der Algorithmus XL, der von Courtois, Klimov, Patarin und Shamir in [CKPS00] vorgestellt wurde, stellt ebenfalls eine Erweiterung der Linearisierungstechnik dar. Dabei steht die Abkürzung XL für *„e**X**tended **L**inearization“* oder auch für *„multipliziere (**X**) und linearisiere (**L**)“*.

Für diesen Algorithmus wird das gegebene, quadratische Gleichungssystem durch Polynome beschrieben, also in der Form

$$\begin{aligned} p_1(x_1, \ldots, x_n) &= 0 \\ \vdots &\quad \vdots \\ p_m(x_1, \ldots, x_n) &= 0 \end{aligned}$$

mit m multivariaten, quadratischen Polynomen $p_1, \ldots, p_m$ in n Variablen. Dabei ist es für XL unerheblich, ob die p_i auch lineare Terme enthalten.
Wie schon in Kapitel 3 sind also die Elemente der Varietät $V_K(\mathfrak{a})$ des von $p_1, \ldots, p_m$ erzeugten Ideals $\mathfrak{a} := (p_1, \ldots, p_m)$ gesucht.

Um den Algorithmus beschreiben zu können, werden zunächst noch einige Schreibweisen benötigt. Da nur Lösungen in K gesucht werden, seien dabei — ähnlich wie in den Abschnitten 1.3.1 und 3.4.1 — alle vorkommenden Polynome immer als modulo $x_i^q - x_i$ reduziert zu betrachten.

Definition 4.2.12

- *Ein Polynom der Form*

$$\prod_{j=1}^{k} x_{i_j} \cdot p_i \quad \textit{mit } 1 \leq i \leq m,\ i_1, \ldots, i_k \in \{1, \ldots, n\}$$

  *heißt **vom Typ** $x^k p$.*

- *Für $D \in \mathbb{N}$ bezeichne $\mathfrak{a}_D$ den Vektorraum, der von Polynomen vom Typ $x^k p$ mit Grad $\leq D$ erzeugt wird, d.h.*

$$\mathfrak{a}_D = \left\langle \prod_{j=1}^{k} x_{i_j} \cdot p_i \;\middle|\; 0 \leq k \leq D-2,\ 1 \leq i \leq m,\ i_1, \ldots, i_k \in \{1, \ldots, n\} \right\rangle_K \subseteq \mathfrak{a}.$$

Der XL-Algorithmus arbeitet in gewisser Weise ähnlich wie Gröbnerbasis-Algorithmen mit beschränktem Grad (vgl. Abschnitt 3.4.1) und durchsucht die Mengen $\mathfrak{a}_D$ für wachsendes D nach univariaten Gleichungen, also Gleichungen, die nur eine der Variablen, etwa x_i, enthalten. Denn ist eine solche Gleichung gefunden, können ihre Lösungen leicht bestimmt

(siehe Abschnitt 2.3.2) und dann in das gegebene System eingesetzt werden. So wird sukzessive die Anzahl der Variablen reduziert, bis letztendlich alle Lösungen gefunden sind.
Das Suchen nach solchen Gleichungen geschieht bei XL mittels einer Linearisierung. Dazu werden alle Gleichungen vom Typ $x^k p$ mit Grad $\leq D$ betrachtet, wobei die verschiedenen auftretenden Terme als unabhängige Variablen aufgefasst werden. Dieses somit lineare System kann dann mittels einer Gauß-Elimination vereinfacht werden. Werden dabei die neuen Variablen entsprechend einer lexikographischen Termordnung auf den ursprünglichen Termen angeordnet, liefert dieses Vorgehen — sofern genügend linear unabhängige Gleichungen vom Grad $\leq D$ vorhanden sind — eine univariate Gleichung in der bezüglich der betrachteten Termordnung kleinsten Variablen.

Zusammengefasst führt dies zu folgendem Algorithmus:

Algorithmus 4.2.13 (XL)
Eingabe: m Polynome $p_1, \ldots, p_m$ in n Variablen
Ausgabe: alle gemeinsamen Nullstellen von $p_1, \ldots, p_m$ in K^n

XL$(p_1, \ldots, p_m)$:

1. **Multiplizieren**: Für ein $D \in \mathbb{N}$ erzeuge alle Polynome vom Typ $x^k p \in \mathfrak{a}_D$ mit $k \leq D-2$.

2. **Linearisieren:** Betrachte alle vorkommenden Terme als neue, unabhängige Variablen und führe eine Gauß-Elimination bezüglich einer Ordnung durch, die diejenigen Variablen zuletzt eliminiert, die von univariaten Termen in einer bestimmten Variablen (etwa x_1) stammen.

3. **a) Lösen:** Liefert Schritt 2 ein univariates Polynom $p(x_1)$, dann bestimme die Nullstellen von p. Dies sind die möglichen Werte für x_1.
   b) Weitersuchen: Liefert Schritt 2 noch kein solches Polynom, starte erneut in Schritt 1 mit größerem D und füge die neuen Polynome zu den bisherigen hinzu.

4. **Wiederholen:** Setze nacheinander die möglichen Werte für x_1 in die Polynome ein und wiederhole den Prozess mit den so vereinfachten Polynomen.

Zur Korrektheit dieses Algorithmus ist das Folgende zu erwähnen: Nach [Moh01] gibt es zu nulldimensionalen Idealen $(p_1, \ldots, p_m)$, für die auch die projektive Varietät $V^*(p_1^*, \ldots, p_m^*)$ zu den Homogenisierungen $p_1^*, \ldots, p_m^*$ projektiv nulldimensional ist, immer ein $D \in \mathbb{N}$, für das in Schritt 2 ein univariates Polynom gefunden werden kann. Da aber immer nur Lösungen im Grundkörper gesucht werden, kann diese Voraussetzung immer erfüllt werden, etwa wie in Abschnit 3.4.1 durch Hinzufügen der Polynome $x_1^q - x_1, \ldots, x_n^q - x_n$ zu den ursprünglichen Polynomen $p_1, \ldots, p_m$.

Der laufzeitbestimmende Schritt des XL-Algorithmus ist die Gauß-Elimination in Schritt 2. Da alle vorkommenden Terme als unabhängige Variablen aufgefasst und Polynome vom Grad $\leq D$ betrachtet werden, muss eine Gauß-Elimination mit ungefähr $\frac{n^D}{D!}$ Variablen durchgeführt werden. Eine untere Schranke für die Komplexität von XL ist also gegeben durch

$$\left(\frac{n^D}{D!}\right)^{\omega}$$

mit $\omega = 3$ für die übliche Gauß-Elimination und $\omega \approx 2,3766$ für optimierte Algorithmen. Entscheidend für die Komplexität ist also der Grad D, der benötigt wird, damit Schritt 2 ein univariates Polynom liefert.

Leider ist der genaue Zusammenhang von D zu m und n eine offene Frage. In [CKPS00] werden dazu folgende Überlegungen beschrieben:
Die Anzahl der in Schritt 1 des XL-Algorithmus erzeugten Gleichungen vom Typ $x^k p$ mit Grad $\leq D$ beträgt ungefähr $\widetilde{m} = \frac{n^{D-2}}{(D-2)!}m$. Im zweiten Schritt werden die darin vorkommenden Terme als insgesamt ungefähr $\widetilde{n} = \frac{n^D}{D!}$ neue Variablen aufgefasst.
Angenommen, die meisten der $\widetilde{m}$ Gleichungen wären linear unabhängig, dann lieferte Schritt 2 ein univariates Polynom, wenn $\widetilde{m} \geq \widetilde{n}$, also

$$m \geq \frac{n^2}{D\,(D-1)}$$

gilt. Es genügte also, ungefähr $D \geq \frac{n}{\sqrt{m}}$ zu wählen.

Leider ist nicht klar, wie viele der in Schritt 1 erzeugten Gleichungen wirklich linear unabhängig sind. Schon die Relinearisierung stellte sich ja im Nachhinein als weniger effektiv heraus als zunächst angenommen wurde, da viele der betrachteten Gleichungen linear abhängig waren.
Dazu stellt Tzuong-Tsieng Moh in [Moh01] durch Betrachtung der Hilbert-Funktion des Ideals $(p_1, \ldots, p_m)$ fest, dass für das Verhältnis $\frac{\widetilde{m}'}{\widetilde{m}}$ der Anzahl der linear unabhängigen, in Schritt 1 erzeugten Polynome zur Anzahl aller solchen Polynome

$$\frac{\widetilde{m}'}{\widetilde{m}} \;\to\; \frac{1}{m} \quad \text{für} \quad D \;\to\; \infty$$

gilt. Dies hätte also (falls schon $\frac{\widetilde{m}'}{\widetilde{m}} \approx \frac{1}{m}$ für recht kleine D gilt) zur Folge, dass nicht $D \geq \frac{n}{\sqrt{m}}$ sondern $D \geq n$ gewählt werden müsste.
Allerdings ist nicht bekannt, wie schnell $\frac{\widetilde{m}'}{\widetilde{m}}$ konvergiert und die einzigen konkreten Werte, die in [Moh01] angegeben werden, basieren auf einigen Simulationen aus [CKPS00] und zeigen nur, dass für D von 4 bis 16 $\frac{\widetilde{m}'}{\widetilde{m}}$ von ungefähr $0,89$ auf ungefähr $0,39$ monoton fällt.
Aufgrund dieser Beobachtung von Moh ist die Annahme der Autoren von [CKPS00], dass $\widetilde{m}' \approx \widetilde{m}$ für das benötigte D gelte, zwar nicht widerlegt; sie scheint aber doch sehr wahrscheinlich falsch zu sein.

Trotzdem sollen im Folgenden noch die unter dieser Annahme möglichen Laufzeitabschätzungen aus [CKPS00] vorgestellt werden:

- Für den Fall $m \approx n$ genügt es, $D \approx \sqrt{n}$ zu wählen. Für die Laufzeit ergibt sich (mit $\sqrt{n}! \approx \left(\frac{\sqrt{n}}{e}\right)^{\sqrt{n}}$) damit ungefähr

$$\left(\frac{n^{\sqrt{n}}}{\sqrt{n}!}\right)^{\omega} \approx e^{c\cdot\sqrt{n}\left(\frac{\ln(n)}{2}+1\right)},$$

  also eine subexponentielle Laufzeit.

- Ist das System überbestimmt mit $m \geq \epsilon n^2$, genügt es unter der oben beschriebenen Annahme, $D \approx \left\lceil \frac{1}{\sqrt{\epsilon}} \right\rceil$ zu wählen. In diesem Fall ergäbe sich also eine Laufzeit von ungefähr

$$\frac{n^{\omega\left\lceil \frac{1}{\sqrt{\epsilon}} \right\rceil}}{\left\lceil \frac{1}{\sqrt{\epsilon}} \right\rceil!},$$

  d.h. eine polynomielle Laufzeit von ungefährem Grad $\frac{\omega}{\sqrt{\epsilon}}$.

In [CKPS00] werden auch einige experimentelle Ergebnisse vorgestellt. In diesen Experimenten wurde über dem Körper $\mathbb{F}_{127}$ für die Fälle $m = n$, $m = n + 1$ und $m = n + 2$ (allerdings für relativ kleine n) untersucht, wie groß D jeweils gewählt werden muss, um eine Lösung zu bekommen. Dabei wurden folgende Zusammenhänge festgestellt:

Fall	$m = n$	$m = n + 1$	$m = n + 2$
benötigtes D	2^n	n	$\sqrt{n} + C$ (vermutlich)

Diese Ergebnisse deuten ebenfalls an, dass die obige Annahme über die lineare Unabhängigkeit für schwach oder gar nicht überbestimmte Systeme (d.h. für $m = n + 1$ oder $m = n$) falsch ist. Für diese Fälle ergibt sich anscheinend eine exponentielle Laufzeit.
Für den Fall $m = n+2$ deuten die Ergebnisse an, dass vielleicht schon $\sqrt{n}+C$ (mit einer Konstanten C) reichen könnte. Dazu wird allerdings bemerkt, dass die durchgeführten Experimente einen zu kleinen Umfang hätten, um eine sichere Aussage über den Fall $m = n + 2$ treffen zu können.
Es ist also weiterhin fraglich, ob die den Komplexitätsbetrachtungen zugrunde liegende Annahme überhaupt aufrechterhalten werden kann.

Zumindest scheinen die Experimente aber zu zeigen, dass mit steigender Differenz $m - n$ der benötigte Grad D stark abfällt. Dies führt zu der Idee für den im folgenden Abschnitt beschriebenen Algorithmus FXL.

4.2.4 FXL

FXL ist ein Algorithmus, mit dem unter der oben getroffenen Annahme über die Laufzeit von XL auch Gleichungssysteme, die nur schwach oder gar nicht überbestimmt sind, in subexponentieller Laufzeit gelöst werden können.
FXL steht dabei für *„**F**ixieren und **XL**“* und dies beschreibt auch schon die grundlegende Idee: Bei FXL werden ein paar (etwa μ) Variablen auf feste Werte gesetzt (fixiert) und das so entstehende, überbestimmte Gleichungssystem in m Gleichungen mit $n - \mu$ Variablen wird mit XL gelöst.

Algorithmus 4.2.14 (FXL)
Eingabe: m Polynome $p_1, \ldots, p_m$ in n Variablen
Ausgabe: alle gemeinsamen Nullstellen von $p_1, \ldots, p_m$ in K^n

FXL$(p_1, \ldots, p_m)$:

1. Wähle μ so, dass bei XL für m Gleichungen und $n - \mu$ Variablen $D \approx \sqrt{n}$ genügt.
2. Für alle $(a_{n-\mu+1}, \ldots, a_n) \in K^\mu$:

   Setze $x_i = a_i$ für $i = n - \mu + 1, \ldots, n$ in $p_1, \ldots, p_m$ ein.
   Löse das so entstehende System in $n - \mu$ Variablen mit XL.

Bei FXL sind also q^μ Schleifendurchläufe durchzuführen und in jedem Durchlauf ist einmal XL für ein überbestimmtes System durchzuführen. Unter der oben diskutierten Annahme benötigte XL eine Laufzeit von ungefähr $e^{c\sqrt{n}\ln(n)}$, so dass sich für FXL eine ungefähre Laufzeit von

$$q^\mu e^{c\sqrt{n}\ln(n)}$$

ergäbe. Wie groß μ dabei gewählt werden muss, ist allerdings noch eine offene Frage.

4.2.5 Vergleich von XL und Relinearisierung

Es ist naheliegend, den Algorithmus XL mit der Relinearisierung zu vergleichen, da beide Algorithmen Linearisierungen nutzen, um überbestimmte Gleichungssysteme zu lösen.
In [CKPS00] wird dazu ein Theorem bewiesen, dass im Wesentlichen aussagt, dass Gleichungssysteme, die mit einer Relinearisierung vom Grad d gelöst werden können, auch durch XL mit $D = d$ gelöst werden können. In dem Beweis dieses Theorems wird konstruktiv gezeigt, dass die Gleichungen, die bei der Relinearisierung ausgenutzt werden, (nach gewissen Transformationen) auch unter den Gleichungen, die bei XL betrachtet werden, gefunden werden können.
Zudem wird dort aufgrund verschiedener Ergebnisse von Experimenten behauptet, dass XL in der Praxis im Allgemeinen effizienter sei als die Relinearisierung. Außer einzelnen Beispielen, für die dies der Fall ist, wird dort aber dafür nur der Grund angeführt, dass es

bei XL möglich ist, auch ungerade Grade D zu nutzen, während bei der Relinearisierung der Grad d immer gerade gewählt werden muss.

4.3 Auswirkungen für HFE

Abschließend wird nun noch kurz betrachtet, welche Auswirkungen die in diesem Kapitel beschriebenen Algorithmen auf die Wahl der Parameter bei HFE haben.

Diese Auswirkungen betreffen im Wesentlichen die in Abschnitt 2.4 vorgestellten Perturbationen:
Für die Perturbationen „$-$“ und „V“, die zu unterbestimmten Gleichungssystemen führen, ist nur der Algorithmus aus Abschnitt 4.1 relevant. Da dieser aber nur Systeme mit $n \geq m^2 + m$ lösen kann, spricht zumindest von dieser Seite nichts dagegen, wie vorgeschlagen mit $k = 1, 2$ oder $\frac{n}{2}$ zu arbeiten.
Etwas anders ist die Situation bei den Perturbationen „$+$“ und „F“, die zu überbestimmten Systemen führen. Nach der Analyse von XL ist zu vermuten, dass schon kleine Werte für k die Lösung solcher Systeme deutlich vereinfachen, so dass k vorsichtig gewählt werden muss.
Allerdings ist die geschätzte Komplexität für konstantes, kleines k subexponentiell und immer noch so groß, dass solche Wahlen von k HFEF- und HFE$^+$-Systeme vermutlich nicht wirklich schwächen. Auf keinen Fall darf k aber so groß gewählt werden, dass $m \geq \epsilon n^2$ mit $\epsilon \approx 0,1$ gilt, da das resultierende System in diesem Fall mit einer einfachen Grad 4-Relinearisierung gelöst werden könnte.

Weitere, etwas deutlichere Auswirkungen auf HFE haben die hier beschriebenen Algorithmen noch durch spezielle Attacken, bei deren Durchführung unter anderem auch stark überbestimmte Gleichungssysteme zu lösen sind. Diese Attacken werden im folgenden Kapitel beschrieben.

Kapitel 5

Spezielle Attacken

Bislang ist keine Attacke bekannt, mit der ein auf HFE basierendes Kryptosystem effizient gebrochen werden kann. Manche der bisher gefundenen Attacken scheinen zwar subexponentielle Laufzeiten zu haben, sind allerdings bei einer für HFE typischen Parameterwahl (etwa q und n so gewählt, dass $q^n > 2^{80}$) teilweise schlechter oder zumindest kaum besser als eine vollständige Suche.

In diesem Kapitel werden die zwei wichtigsten der momentan bekannten Attacken auf HFE vorgestellt und ihre wesentlichen Aspekte beschrieben. Dabei sind grundsätzlich zwei Arten von Attacken zu unterscheiden:

- Attacken, die versuchen, das HFEP zu lösen, d.h. Attacken die zu festem $y \in K^n$ mindestens eine Lösung $x \in K^n$ mit $P(x) = y$ liefern;
- strukturelle Attacken, d.h. Attacken, die Informationen (wie beispielsweise den geheimen Schlüssel) liefern, die es ermöglichen, P effizient umzukehren.

Zur ersten Art von Attacken sind vor allem die Attacken über implizite Gleichungen zu erwähnen, wie Courtois sie in [Cou01a] beschreibt. Diese werden in Abschnitt 5.1 vorgestellt. Zu dieser Kategorie gehört auch die Attacke über affine Vielfache (siehe Abschnitt 5.1.2), die Patarin schon in [Pat96a] beschreibt und mit der er das Verfahren C^* gebrochen hat.
Zur zweiten, oben erwähnten Art von Attacken zählen vor allem Attacken, die versuchen, die Kenntnis auszunutzen, dass einem HFE-System ein Polynom kleinen Grades zugrunde liegt. Wie schon in Abschnitt 2.5.1 angedeutet, führt dieser Ansatz nach einigen Transformationen zum Problem MinRank. Die wichtigste solche Attacke wurde von Kipnis und Shamir in [KS99] vorgestellt und wird hier zusammen mit einer von Courtois in [Cou01a] vorgeschlagenen Verbesserung in Abschnitt 5.2 beschrieben.

5.1 Attacken über implizite Gleichungen

In diesem Abschnitt wird ein Überblick über die von Courtois in [Cou01a] vorgestellten Attacken auf das HFE-Problem gegeben. Diese Attacken liefern nicht den geheimen Schlüssel, sondern können im Allgemeinen nur zu einem vorgegebenen Wert y ein x mit $P(x) = y$ liefern.

In Abschnitt 5.1.1 werden zunächst die grundlegenden Ideen dieser Attacken vorgestellt. Im folgenden Abschnitt 5.1.2 wird dann als Beispiel das Wesentliche über die erste Attacke dieser Art, die Attacke über affine Vielfache, beschrieben. Zuletzt wird in Abschnitt 5.1.3 dann noch ein Überblick über Verallgemeinerungen und Verbesserungen von Attacken dieser Art gegeben, wie Courtois sie in [Cou01a] vorstellt.

5.1.1 Idee

Bei Public-Key-Kryptosystemen, die auf einer Einwegfunktion $P = (p_1, \ldots, p_n)$ beruhen, ist es allgemein möglich, diese Funktion in Form von Relationen

$$y_i = p_i(x_1, \ldots, x_n)$$

zwischen den Eingabewerten $x_1, \ldots, x_n$ und den Ausgabewerten $y_1, \ldots, y_n$ zu beschreiben. Eine Attacke auf ein solches System hat das Ziel, möglichst einfache Relationen zwischen den x_i und den y_i zu finden, aus denen sich nach Einsetzen von konkreten Werten für die y_i die x_i berechnen lassen.
Solche Relationen lassen sich immer durch ein Polynom $\omega \in K[x_1, \ldots, x_n, y_1, \ldots, y_n]$ mit

$$\omega(a_1, \ldots, a_n, p_1(a_1, \ldots, a_n), \ldots, p_n(a_1, \ldots, a_n)) = 0 \qquad \text{für alle } (a_1, \ldots, a_n) \in K^n$$

beschreiben. Beispielsweise ist eine Beschreibung der durch die Funktion P vorgegebenen, oben erwähnten Relationen immer durch $\omega := y_i - p_i(x_1, \ldots, x_n)$ gegeben.

Es stellt sich die Frage, wann solche Relationen als einfach zu bezeichnen sind.
Courtois schlägt in [Cou01a] vor, dazu den Grad der Relationen als Funktionen in x (d.h. insbesondere modulo $x_i^{q_i} - x_i$ reduziert) zu betrachten. Denn nach Einsetzen eines konkreten $y \in K^n$ liefern solche Relationen im Allgemeinen ein polynomiales Gleichungssystem für die x_i von eben diesem Grad. Dieses System ist natürlich umso einfacher zu lösen, je kleiner der Grad ist.

Zudem unterscheidet Courtois zumindest informal zwischen trivialen und nicht-trivialen Gleichungen, wobei er als triviale Gleichungen diejenigen Relationen bezeichnet, die sich als einfache Kombinationen der ursprünglichen Gleichungen $y_i = p_i(x_1, \ldots, x_n)$ ergeben und deren Grad nach Einsetzen eines konkreten $y \in K^n$ nicht kleiner wird.

Beispiel 5.1.1
Sei $K = \mathbb{F}_2$. Etwa zu

$$\begin{aligned} y_1 &= x_1x_2 + x_2x_3 + x_2 + x_3 \\ y_2 &= x_2x_3 + x_1 + x_4 \\ y_3 &= x_2x_4 + x_1 + x_3 \end{aligned}$$

ist dann

$$y_1 + y_2 = x_1x_2 + x_1 + x_2 + x_3 + x_4$$

eine triviale Relation, da diese Gleichung nach Ersetzen von y_1 und y_2 durch konkrete Werte aus $\mathbb{F}_2$ immer noch den Grad 2 in den x_i hat. Dagegen ist

$$\underbrace{y_1 + x_2y_2 + y_3}_{=x_2^2x_3+x_2x_3+x_2+x_1} = x_1 + x_2$$

eine nicht-triviale Gleichung, die zu festen Werten von y_1, y_2 und y_3 eine lineare Beziehung zwischen x_1 und x_2 liefert.

Triviale Gleichungen sind also die Gleichungen, die direkt aus den öffentlich bekannten Relationen $y_i = p_i(x_1, \ldots, x_n)$ konstruiert werden können und nur Gleichungen des ohnehin schon vorhandenen (oder eines höheren) Grades liefern. Die Konstruktion solcher Gleichungen ist immer leicht möglich, bringt einem Angreifer aber keine wirklichen Vorteile gegenüber den Ausgangsrelationen.
Nicht-triviale Gleichungen dagegen stellen eine Vereinfachung der bekannten Relationen dar und geben einem Angreifer zusätzliche Informationen über die gesuchten Werte.

Ziel einer sinnvollen Attacke sollte es also sein, nicht-triviale Gleichungen zu finden. Dafür gibt es im Wesentlichen zwei Ansätze:
Solche Gleichungen könnten explizit konstruiert werden, etwa indem die bekannten Gleichungen geschickt miteinander kombiniert werden. Dies ist beispielsweise bei verschiedenen Eliminationsstrategien, wie Gröbnerbasis-Algorithmen oder auch den in Kapitel 4 vorgestellten Algorithmen, der Fall.
Die andere Möglichkeit (nämlich gerade der Ansatz über implizite Gleichungen) sieht vor, die Koeffizienten solcher nicht-trivialen Gleichungen als Variablen zu betrachten. Aus den Relationen $y_i = p_i(x_1, \ldots, x_n)$ bzw. aus bekannten Klartext-Chiffretext-Paaren $(x, P(x))$ können dann nämlich Bedingungen für diese Koeffizienten erhalten werden, die es ermöglichen, nicht-triviale Gleichungen zu bestimmen.

Die erste Attacke dieser Art war die Attacke über affine Vielfache, mit der Patarin in [Pat95] das Verfahren C^* gebrochen hat. Bevor nun in Abschnitt 5.1.2 anhand dieses Beispiels der genaue Ablauf einer solchen Attacke beschrieben wird, werden vorher noch

Kurzschreibweisen für verschiedene Typen von Gleichungen vereinbart, wie Courtois sie in [Cou01a] einführt:

Notation 5.1.2

1. *Ein* ***Gleichungstyp*** *ist eine Vereinigung von Termen in den formalen Variablen x, y, X und Y.*

2. *Ein Ausdruck $x^k y^l$ umfasst alle Terme $x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \cdot y_1^{j_1} \cdot \ldots \cdot y_n^{j_n}$ mit*

$$\sum i_\nu = k, \quad \sum j_\nu = l \quad \text{und} \quad i_\nu, j_\nu < q.$$

3. *Ein Ausdruck $X^k Y^l$ ist definiert als Vereinigung aller Terme mit kleinerem oder gleichem Grad:*

$$X^k Y^l := \bigcup_{\substack{0 \le k' \le k \\ 0 \le l' \le l}} x^{k'} y^{l'}.$$

4. *Mit $\{Gl.Typ\}$ wird die Menge der Terme bezeichnet, die im Gleichungstyp $Gl.Typ$ verwendet werden und mit $[Gl.Typ]$ wird die Menge der Gleichungen bezeichnet, die daraus gebildet werden kann.*

Beispiel 5.1.3
$XY \cup x^2 = 1 \cup x \cup y \cup xy \cup x^2$ ist ein Gleichungstyp, wobei $\left[XY \cup x^2\right]$ alle Gleichungen der Form

$$\sum \alpha_{ij} x_i x_j + \sum \beta_{ij} x_i y_j + \sum \gamma_i x_i + \sum \delta_i y_i + \mu_0 = 0$$

mit α_{ij}, β_{ij}, γ_i, δ_i, $\mu_0 \in K$ umfasst.

5.1.2 Attacke über affine Vielfache

Die Attacke über affine Vielfache ist eine Attacke, die versucht, Gleichungen vom Typ XY^l für möglichst kleine l zu finden. Diese spezielle Art einer Attacke über implizite Gleichungen wird in [Fel01] ausführlich analysiert. Deshalb werden hier nur die wesentlichen Aspekte dargestellt, die nötig sind, um den allgemeinen Ablauf einer Attacke über implizite Gleichungen an diesem Beispiel zu verdeutlichen.

In [Fel01] wird gezeigt, dass zu jedem Polynom, das einem HFE-System zugrunde liegt, ein sogenanntes affines Vielfaches existiert. Weiter wird gezeigt, dass ein solches affines Vielfaches die Existenz einer gewissen Anzahl von Relationen vom Typ XY^l garantiert. Dabei beschreibt l gerade das maximale q-Gewicht eines Exponenten bei y in diesem affinen Vielfachen.
Beispielsweise für C^* kann sogar gezeigt werden, dass es immer ein affines Vielfaches mit $l = 1$ gibt, dass für C^* also immer Gleichungen vom Typ XY existieren. Diese

Feststellung bildet die Grundlage für die in Abschnitt 2.1 skizzierte Attacke, mit der Patarin C^* gebrochen hat und die in [Fel01] ausführlich beschrieben wird.

Im Folgenden wird der allgemeine Ablauf einer solchen Attacke in groben Zügen beschrieben:
Angenommen, es existieren Relationen zwischen den x_i und den y_i vom Typ XY^l, also Gleichungen der Form

$$\sum_{k=1}^{l}\left(\sum_{\substack{j_1,\ldots j_n\\ \sum j_\nu=k}}\left(\sum_{i=1}^{n}\alpha_{ij_1\ldots j_n}\cdot x_i y_1^{j_1}\cdot\ldots\cdot y_n^{j_n}\right)+\beta_{j_1\ldots j_n}\cdot y_1^{j_1}\cdot\ldots\cdot y_n^{j_n}\right)+\sum\gamma_i x_i+\mu_0=0$$

mit $\alpha_{ij_1\ldots j_n}$, $\beta_{j_1\ldots j_n}$, γ_i, $\mu_0\in K$. Dann sind diese Gleichungen für beliebige Klartext-Chiffretext-Paare $(x,y)=(a,P(a))$ mit $a\in K^n$ erfüllt.
Werden in der obigen, allgemeinen Form die $\alpha_{ij_1\ldots j_n}$, $\beta_{j_1\ldots j_n}$, γ_i und μ_0 als Variablen aufgefasst und Werte der Form $(a,P(a))$ für $(x_1,\ldots,x_n,y_1,\ldots,y_n)$ eingesetzt, so liefert dieses Vorgehen also lineare Bedingungen für die Koeffizienten der gesuchten Relationen. Werden *genügend* solche Paare eingesetzt, sind die Koeffizienten $\alpha_{ij_1\ldots j_n}$, $\beta_{j_1\ldots j_n}$, γ_i und μ_0 durch diese Bedingungen so stark eingeschränkt, dass nur noch Relationen übrig bleiben, die für alle Paare $(a,P(a))$ (und nicht nur für die zufällig gewählten und hier eingesetzten) erfüllt sind.
Dabei ist es allerdings sehr schwierig zu beurteilen, was in diesem Zusammenhang eine *genügende* Anzahl ist (vgl. auch [Fel01]).

Soll nun für ein bestimmtes $y\in K^n$ ein x mit $P(x)=y$ gefunden werden, so kann dieses y einfach in die so erhaltenen Relationen eingesetzt werden. Dies liefert ein lineares Gleichungssystem für die x_i, das die möglichen Werte für x zumindest stark einschränkt oder sogar eindeutig liefert.

Der Aufwand einer solchen Attacke ist vor allem von der Anzahl der in den Relationen auftretenden Koeffizienten abhängig, da diese durch eine Gauß-Elimination bestimmt werden. Im hier beschriebenen Fall ist diese Anzahl (abgesehen von den möglichen Beschränkungen der j_i durch q) gerade gleich

$$(n+1)\binom{n+l}{l}\approx O\left(n^{l+1}\right).$$

Deshalb ist es für das Gelingen dieser Attacke so wichtig, dass ein affines Vielfaches mit „kleinem l“ (siehe unten) existiert.
Andersherum betrachtet, ist es eben aus diesem Grunde für einen Erzeuger eines HFE-Systems wichtig, als Grundlage für das System ein φ zu wählen, für das kein affines Vielfaches mit „kleinem l“ existiert. Genauer gesagt sollten nur affine Vielfache existieren, für die ungefähr $l>\frac{80}{3\log_2 n}-1$ erfüllt ist, denn für diese ist $\left(n^{l+1}\right)^3>2^{80}$, die Gauß-Elimination auf den n^{l+1} Variablen also nicht mehr durchführbar.

5.1.3 Weitere Attacken

Das Vorgehen bei der Attacke über affine Vielfache, das im letzten Abschnitt beschrieben wurde, kann weitgehend verallgemeinert werden, wie Courtois es in [Cou01a] beschreibt. Bei der Attacke über affine Vielfache wurden Gleichungen vom Typ XY^l vor allem deshalb betrachtet, weil durch die Existenz eines affinen Vielfachen auch die Existenz einer gewissen Anzahl solcher Gleichungen gesichert ist (vgl. [Fel01]). Eine mögliche Verallgemeinerung ist nun, auch Gleichungen anderer Typen als XY^l zu untersuchen. Für die im Folgenden betrachteten Gleichungstypen ist dann allerdings die Existenz der gesuchten Gleichungen wenn überhaupt nur experimentell bestätigt.

Zunächst werden zwei Arten von Gleichungstypen charakterisiert, die für Attacken über implizite Gleichungen von besonderer Bedeutung sind:

Definition 5.1.4
Ein Gleichungstyp wird als

- ***invariant*** *bezeichnet, falls die Menge der zugehörigen Terme invariant ist unter Variablentransformationen, die durch bijektive, affine Transformationen beschrieben werden,*
- y***-dominant*** *bezeichnet, falls die zugehörigen Gleichungen nach Substitution mit $y = 0$ vom Typ X, also affin in den x_i, sind.*

Beispielsweise beschreibt X^2Y einen invarianten Gleichungstyp, während x^2y nicht invariant ist, da Gleichungen von diesem Typ etwa durch eine Transformation der Form $x \mapsto x + 1$ in Gleichungen vom Typ $x^2y \cup xy \cup y$ verwandelt werden.
Das Suchen nach Gleichungen von einem invarianten Gleichungstyp hat den Vorteil, dass zwar nicht die Gleichungen selber, aber doch die Anzahl der zu findenden Gleichungen vom für y einzusetzenden Wert unabhängig ist. Kurz gesagt liegt dies daran, dass das Einsetzen verschiedener Werte für y durch verschiedene bijektive, affine Transformationen des ursprünglichen Systems und anschließendes Einsetzen von $y = 0$ sozusagen simuliert werden kann.
Dadurch kann also ohne Einschränkung angenommen werden, dass Lösungen für $P(x) = 0$ gesucht werden. Dieses Vorgehen kann auch praktisch umgesetzt werden, indem zunächst P entsprechend affin transformiert wird, dann Lösungen x für $y = 0$ bezüglich des neuen Systems bestimmt werden und schließlich diese x zurücktransformiert werden.

Die y-dominanten Gleichungstypen, wie beispielsweise $XY \cup x^2y \cup xy^2 \cup x^3y$ haben den Vorteil, dass sie nach Einsetzen des konkreten Wertes $y = 0$ — ähnlich wie in der Attacke über affine Vielfache — ein lineares System für die x_i liefern, das dann auf jeden Fall leicht zu lösen ist. Ein weiterer Vorteil solcher Attacken ist, dass die Koeffizienten der Relationen

nicht unbedingt vollständig berechnet werden müssen, da durch die Substitution mit $y = 0$ sowieso ein Großteil der Koeffizienten ausgelöscht wird.
Aber Attacken mit y-dominanten Gleichungen, die erst nach Substitution mit $y = 0$ affin in den x_i werden, haben auch einen gewissen Nachteil. Denn eine solche Attacke muss für jedes y, für das $P(x) = y$ zu lösen ist, komplett neu durchgeführt werden, da, um $y = 0$ einsetzen zu können, zunächst P abhängig von y transformiert werden muss. Dies unterscheidet sie von der Attacke über affine Vielfache, die mit Gleichungen vom Typ XY arbeitet und somit für beliebige Werte, die für y substituiert werden (und nicht nur für $y = 0$), ein lineares System für die x_i liefert.

Der wesentlichen Aspekt der Komplexität einer solchen Attacke ist — wie schon bei der Attacke über affine Vielfache zu sehen war — die Größe der Gleichungen des betrachteten Typs, da diese angibt, wie viele Koeffizienten durch die Gauß-Elimination bestimmt werden müssen. Courtois verweist für diese Größen in [Cou01a] auf eine vollständige Übersicht in seiner (noch nicht erschienenen) Dissertation [Cou01c], die es ermöglichen soll, diese Größen zu berechnen.

Damit eine solche Attacke über implizite Gleichungen funktionieren kann, ist es wichtig, dass es überhaupt nicht-triviale Gleichungen vom untersuchten Typ gibt. Dazu gibt Courtois in [Cou01a] eine Übersicht für $n = 21$ an, in der für verschiedene Gleichungstypen und verschiedene Grade d die Anzahl der gefundenen nicht-trivialen Gleichungen aufgelistet wird. Leider wird dabei aber nicht genauer spezifiziert, wie die angegebenen Werte berechnet wurden.
Interessant an diesen Werten ist vor allem die Beobachtung, dass für $d = q^k + 1, \ldots, q^{k+1}$ jeweils ähnliche Werte gefunden wurden, so wie dies auch bei den für diese Arbeit gemessenen Laufzeiten der Gröbnerbasisberechnungen in Abschnitt 3.4.2 der Fall war.
Courtois stellt aufgrund der berechneten Werte folgende Vermutung auf:

Vermutung 5.1.5
Zu einem HFE-System, das auf einer versteckten Funktion vom Grad d beruht, gibt es $O(n)$ nicht-triviale Gleichungen vom Typ

$$\left[X \cup x^2 y \cup \ldots \cup x^{\frac{1}{2}\left\lceil \log_q d \right\rceil - 1} y\right].$$

Ebenfalls aus diesen Berechnungen für $n = 21$ heraus, extrapoliert er folgende Schätzungen der Komplexität bestimmter Attacken:
Etwa für $n = 64$ und $d \leq 24$ erhält er aus seinen Berechnungen eine Attacke mit Gleichungen vom Typ $XY \cup x^2 y \cup xy^2$, die mit einem Platzbedarf von 8 Gigabyte und einer Laufzeit von ungefähr 2^{48} Taktzyklen, also wenigen Tagen auf einem PC, auskommt. Somit ist HFE für $d \leq 24$ nicht sicher.

Für die HFE-Challenge 1 aus [Pat96a], ein konkretes HFE-System mit $n = 80$ und $d = 96$, gibt er für eine Attacke mit Gleichungen vom Typ $XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$ einen Platzbedarf von 33 Terabyte und eine Laufzeit von 2^{62} Taktzyklen an, also eine eher unrealistische, nicht durchführbare Attacke.

Um zumindest den Platzbedarf solcher Attacken zu verringern, schlägt er dann noch weitere, verbesserte Versionen vor:
Sein erster Vorschlag, die sogenannte „Reconciliation Technique“, sieht vor, eine feste Teilmenge der x_i auf 0 zu fixieren und nur die anderen zufällig zu wählen. Dies reduziert die Anzahl der zu berechnenden Koeffizienten und somit den Platzbedarf erheblich. Nach Courtois ist es dann möglich, die nicht-trivialen Gleichungen aus mehreren solchen Berechnungen, bei denen verschiedene Teilmengen der x_i auf 0 gesetzt wurden, geschickt zu kombinieren, um allgemein gültige Gleichungen zu erhalten.
Allerdings treten bei dieser Technik noch weitere, sogenannte künstliche Gleichungen auf, die sich nur vermeiden lassen, indem die Anzahl der auf 0 gesetzten x_i stark beschränkt wird. Um mehr x_i auf 0 setzen zu können, verwendet Courtois die sogenannte „Distillation Technique“, eine Verfeinerung der „Reconciliation Technique“, die ebenfalls in seiner Dissertation [Cou01c] genauer vorgestellt werden soll.
Mit Hilfe dieser Techniken kommt er dann für die oben erwähnte Attacke auf die HFE-Challenge 1 auf einen Platzbedarf von „nur noch“ 390 Gigabytes bei einer Laufzeit von 2^{62} Taktzyklen, also eine Attacke, die — wenn diese Schätzungen stimmen — am Rande des mit sehr großem Aufwand gerade noch Durchführbaren liegt.

Insgesamt scheinen diese Attacken über implizite Gleichungen eine interessante Möglichkeit darzustellen, multivariate Kryptosysteme zu attackieren. Allerdings ist die Frage der wirklichen Komplexität dieser Attacken noch offen und die bisher von Courtois angedeuteten Techniken scheinen für HFE und vor allem für HFE-Varianten (vgl. Abschnitt 2.4) noch keine effizient durchführbaren Attacken zu liefern.

5.2 Attacken über MinRank

Wie schon in Abschnitt 2.5.1 erwähnt, ist es möglich, auszunutzen, dass ein HFE-System auf einer versteckten Funktion φ von kleinem Grad d beruht, um das Finden des geheimen Schlüssels auf das Problem MinRank zurückzuführen. In diesem Abschnitt wird nun zusammengefasst, wie Kipnis und Shamir in [KS99] diese Reduktion auf MinRank durchgeführt haben und wie sie dann versucht haben, das Problem MinRank möglichst effizient zu lösen. Außerdem wird ein Verbesserungsvorschlag von Courtois aus [Cou01a] vorgestellt, der die MinRank-Lösung beschleunigt.

Um die Attacke von Kipnis und Shamir so durchführen zu können, wie sie in [KS99] beschrieben ist, müssen zunächst einige Einschränkungen an das betrachtete HFE-System getroffen werden. Dort werden nämlich nur HFE-Systeme betrachtet, die auf einem HFE-Polynom der Form

$$f(x) = \sum_{i,j=0}^{r-1} \beta_{ij} x^{q^i+q^j} \in L[x]$$

ohne rein lineare oder absolute Summanden beruhen. Zudem werden nur rein lineare Transformationen s und t betrachtet.
Dann enthalten die multivariaten Darstellungen sowohl der öffentlichen Funktion P als auch die von φ_{K^n} nur quadratische Terme. (In Charakteristik 2 enthalten sie streng genommen auch lineare Terme, die aber wegen $x_i^2 = x_i$ auch als quadratische Terme geschrieben werden können.)
Der Grad d des HFE-Polynoms wird in diesem Abschnitt durch den Parameter r mit $d \leq 2q^{r-1}$ bestimmt.

Nach den Sätzen aus Abschnitt 1.3.1 kann zu einer Funktion $P : K^n \to K^n$ mit $M(P) \in \mathcal{M}_{n,2}$ (etwa der öffentlichen Funktion P eines HFE-Systems) immer ein univariates HFE-Polynom $U(P) \in \mathcal{U}_{n,2}$ gefunden werden. Im Allgemeinen kann dieses Polynom nur deshalb nicht zur Umkehrung von P verwendet werden, da sein Grad meist (d.h. mit Wahrscheinlichkeit $\frac{q^n-1}{q^n}$) $2q^{n-1}$ beträgt.

Die Idee der Attacke von Kipnis und Shamir ist, Transformationen s und t zu suchen, für die $U\left(t^{-1} \circ P \circ s^{-1}\right)$ einen deutlich kleineren Grad als $2q^{n-1}$ (beispielsweise etwa d) hat. Solche Transformationen existieren nach der Konstruktion eines HFE-Systems in jedem Fall. Es ist aber noch nicht einmal notwendig, genau die ursprünglichen, bei der Konstruktion verwendeten Transformationen (also den geheimen Schlüssel) zu finden, sondern es genügt, irgendwelche Transformationen mit der beschriebenen Eigenschaft zu finden.

Reduktion auf MinRank

Dazu wird die Gradbedingung auf das schon in Abschnitt 2.5.1 eingeführte Problem MinRank zurückgeführt. Da MinRank allerdings für Matrizen formuliert ist, wird zunächst erläutert, wie die hier auftretenden Abbildungen durch Matrizen beschrieben werden können:

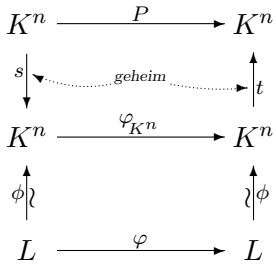
Sei $\Psi : K^n \to K^n$ eine Abbildung mit $M(\Psi) \in \mathcal{M}_{n,2}$, so dass in der multivariaten Darstellung $M(\Psi)$ keine linearen oder absoluten Terme auftreten. Dann ist aus dem Beweis von Satz 1.3.9 ersichtlich, dass $U(\Psi) \in \mathcal{U}_{n,2}$ von der Form $\sum_{i,j=0}^{n-1} u_{ij} x^{q^i+q^j} =: u$ sein muss, also ebenfalls keine linearen oder absoluten Anteile enthält.

Somit kann die auf L übertragene Abbildung $\Psi_L := \phi^{-1} \circ \Psi \circ \phi$ (vgl. Abschnitt 1.3.1) durch die Matrix $M_\Psi := (u_{ij})_{0 \le i,j \le n-1} \in L^{n \times n}$ dargestellt werden als

$$\Psi_L(x) = u(x) = \sum_{i,j=0}^{n-1} u_{ij} x^{q^i+q^j} = \underline{x} M_\Psi \underline{x}^t \quad \text{mit } \underline{x} = \left(x^{q^0}, x^{q^1}, \ldots, x^{q^{n-1}}\right).$$

Dies ist zwar eine ungewöhnliche Darstellung einer quadratischen Form, aber, wie unten zu sehen ist, ermöglicht sie gerade, die Einflüsse der Transformationen s und t auf Ψ präzise zu erfassen.

Zur Erinnerung noch einmal das allgemeine HFE-Schema:



Wie oben schon beschrieben, werden in diesem Abschnitt nur HFE-Systeme betrachtet, für die die multivariaten Darstellungen von P und φ_{K^n} keine linearen oder absoluten Terme enthalten. Die im obigen Schema vorkommenden P und φ_{K^n} sind also genau von der Form, dass sie, wie gerade erläutert, durch Matrizen $G := M_P$ und $H := M_{\varphi_{K^n}}$ beschrieben werden können.

Für ein öffentliches HFE-System P ist die Matrix G effizient berechenbar, beispielsweise wie in Satz 1.3.9 angedeutet oder auch einfach durch Interpolation über einige Paare $(x, P(x))$.
Die Matrix H ist zunächst unbekannt. Allerdings ist aufgrund des kleinen Grades d des HFE-Polynomes ($d \le 2q^{r-1}$) bekannt, dass mit $H = (h_{ij})_{0 \le i,j \le n-1}$

$$h_{ij} = 0 \quad \text{für } (i \ge r \text{ oder } j \ge r)$$

gilt. Es muss also $Rang(H) \le r$ gelten.

Da die Transformationen s und t nach Voraussetzung nur als linear (und nicht als affin) angenommen werden, sind auch die Abbildungen $t^{-1} \circ P$ und $\varphi_{K^n} \circ s$ des HFE-Schemas durch solche Matrizen darstellbar. Insbesondere ist in dieser Matrixdarstellung der Einfluss der Transformationen besonders gut sichtbar, wie der folgende Satz zeigt, den Kipnis und Shamir in [KS99] bewiesen haben:

Satz 5.2.1

Seien $s_0, \ldots, s_{n-1}, t_0, \ldots, t_{n-1} \in L$ die Koeffizienten der univariaten Darstellung der linearen Transformationen $s : K^n \to K^n$ und $t^{-1} : K^n \to K^n$, d.h.

$$U(s) = \sum_{i=0}^{n-1} s_i x^{q^i} \qquad \text{und} \qquad U\left(t^{-1}\right) = \sum_{i=0}^{n-1} t_i x^{q^i}.$$

Seien weiter für die zu P gehörende Matrix $G = (g_{ij})$ Matrizen G^{*k} für $k = 0, \ldots, n-1$ durch

$$G^{*k} := \left(\left(g_{i-k,j-k}\right)^{q^k} \right)_{0 \le i,j \le n-1}$$

und zur Transformation s die Matrix

$$W := \left(\left(s_{j-i}\right)^{q^i} \right)_{0 \le i,j \le n-1}$$

definiert, wobei die Indizes jeweils modulo n zu lesen sind.
Dann können die Abbildungen $t^{-1} \circ P$ und $\varphi_{K^n} \circ s$ dargestellt werden durch die Matrizen

$$M_{t^{-1} \circ P} = \sum_{k=0}^{n-1} t_k G^{*k} \qquad \text{und} \qquad M_{\varphi_{K^n} \circ s} = WHW^t.$$

Nach Konstruktion eines HFE-Schemas gilt $P = t \circ \varphi_{K^n} \circ s$ und damit auch $t^{-1} \circ P = \varphi_{K^n} \circ s$, d.h. auch die beiden Matrizen aus dem vorigen Satz müssen übereinstimmen:

$$WHW^t = \sum_{k=0}^{n-1} t_k G^{*k} \quad =: G'.$$

Damit ist aber nun die Reduktion auf MinRank (vgl. Problem 2.5.4) gelungen: Die Matrizen G^{*k} können direkt aus dem durch P bekannten G berechnet werden. Zudem ist wegen $Rang(H) \le r$ natürlich auch $Rang(G') \le r$.
Es genügte also ein effizienter Algorithmus für das Problem MinRank, um die Koeffizienten $t_0, \ldots, t_{n-1}$, die die Transformation t festlegen, zu bestimmen.
Leider ist ein solcher, allgemeiner Algorithmus nicht bekannt. Vielmehr ist das MinRank-Problem im Allgemeinen für endliche Körper $\mathcal{NP}$-hart.
Allerdings sind zwei Ansätze bekannt, wie das hier auftretende Problem gelöst werden kann, wenn r nicht zu groß wird. Diese werden nun vorgestellt:

Bestimmung von t

In [KS99] schlagen Kipnis und Shamir folgendes Vorgehen zur Bestimmung der t_i vor:
Sei $K_{G'} := \{x \in K^n \mid xG' = 0\}$ der linke Kern zur Matrix G'. Wegen $Rang(G') \le r$ ist dann $\dim_K(K_{G'}) \ge n - r$.

Kipnis und Shamir behaupten nun, dass damit zu erwarten sei, dass in $K_{G'}$ $n-r$ linear unabhängige Vektoren $x_1, \ldots, x_{n-r}$ gefunden werden können, sogar wenn die ersten $n-r$ Stellen dieser Vektoren auf beliebige Werte festgelegt werden. Diese Behauptung ist vermutlich so zu verstehen, dass diese „beliebigen“ Werte so gewählt werden, dass die Vektoren auf jeden Fall linear unabhängig sind.
Die restlichen r Stellen der $n-r$ Vektoren werden als neue Variablen aufgefasst. Jede Vektorgleichung der Form $x_i G' = 0$ liefert somit n skalare, quadratische Gleichungen über L in den Variablen $t_0, \ldots, t_{n-1}$ und den zum Vektor x_i neu definierten Variablen. Insgesamt ergibt sich so ein quadratisches Gleichungssystem mit $n\,(n-r)$ Gleichungen in $r\,(n-r)+n$ Variablen über L.
Im Grunde genommen wurde also die Lösung des ursprünglichen Problems, das quadratische Gleichungssystem $P\,(x) = y$ über K zu lösen, auf das — auf den ersten Blick schwierigere — Problem zurückgeführt, dieses neue, ebenfalls quadratische Gleichungssystem über dem größeren Körper L zu lösen. Der Vorteil dieses neuen Systems besteht aber darin, dass es überbestimmt ist, in den Begriffen von Kapitel 4 mit

$$\epsilon = \frac{n\,(n-r)}{(r\,(n-r)+n)^2} \approx \frac{1}{r^2}.$$

Kipnis und Shamir stellen an dieser Stelle nun die Relinearisierungstechnik vor und kommen, wie in Kapitel 4 beschrieben, zu dem Schluss, dass mit dieser Technik für gebräuchliche n und r die hier auftretenden Systeme gelöst werden können. Allerdings ist die hier auftretende Überbestimmtheit $\epsilon \approx \frac{1}{r^2}$ mit $r \approx \log_q d$ schon ziemlich klein, so dass, wie in Kapitel 4 gezeigt, Relinearisierungen mit deutlich größerem Grad durchgeführt werden müssen, als Kipnis und Shamir es ursprünglich annahmen.

Ein alternatives Vorgehen zur Lösung des MinRank-Problems schlägt Courtois in [Cou01a] vor:
Da $Rang\,(G') \leq r$ ist, muss jede $(r+1) \times (r+1)$-Untermatrix von G' singulär sein. Zu jeder solchen Untermatrix U muss also $\det(U) = 0$ gelten, wobei $\det(U)$ als Polynom vom Grad $r+1$ in den Variablen $t_0, \ldots, t_{n-1}$ über L geschrieben werden kann. Die Rangbedingung für G' kann damit ausgedrückt werden durch ein polynomiales Gleichungssystem mit $\binom{n}{r+1}^2$ Gleichungen vom Grad $r+1$ in den n Variablen $t_0, \ldots, t_{n-1}$. Wären zumindest $\binom{n}{r+1}$ dieser Gleichungen linear unabhängig, könnte das System durch eine Linearisierung gelöst werden, da in den Gleichungen vom Grad $r+1$ ungefähr $\binom{n}{r+1}$ verschiedene Terme auftreten. Es wäre dann also ein System der Größe $\binom{n}{r+1} \approx n^{r+O(1)} \approx n^{\log_q d + O(1)}$ zu lösen, um die $t_0, \ldots, t_{n-1}$ zu bestimmen.

Bestimmung von s

Ist es auf eine dieser beiden Weisen gelungen, die Transformation t zu bestimmen, so ist die Matrix G' nun bekannt. Dadurch ist es vergleichsweise einfach möglich, auch s zu

bestimmen:
Dazu sei angenommen, dass $Rang(W) = n$ und $Rang(H) = r$ ist. Da $h_{ij} = 0$ für $i \geq r$ gilt, folgt dann

$$Kern(H) = \{x \in K^n \mid xH = 0\} = \{x = (x_0, \ldots, x_{n-1}) \in K^n \mid x_j = 0 \text{ für } 0 \leq j < r\}.$$

Wegen $Rang(W) = n$ folgt zudem

$$\begin{aligned} Kern\left(G'\right) &= Kern\left(WHW^t\right) = Kern(WH) \\ \text{und} \quad Rang\left(G'\right) &= Rang\left(WHW^t\right) = Rang(H) = r. \end{aligned}$$

Also existiert eine Basis $v_1, \ldots, v_{n-r}$ von $Kern(G')$, für die zudem auch $v_i W \in Kern(H)$ für $i = 1, \ldots, n-r$ ist. Nach der obigen Beschreibung von $Kern(H)$ liefert dies also $(n-r)\,r$ Gleichungen der Form

$$(v_i W)_j = 0 \qquad \text{für } i = 1, \ldots, n-r \text{ und } j = 0, \ldots, r-1$$

für die n^2 Einträge von W, also ein eigentlich unterbestimmtes Gleichungssystem. Allerdings ist die Matrix W in Satz 5.2.1 definiert als

$$W := \left((s_{j-i})^{q^i}\right)_{0 \leq i,j \leq n-1},$$

d.h. nur abhängig von den n Variablen $s_0, \ldots, s_{n-1}$. Eingesetzt in die Gleichungen $(v_i W)_j = 0$ ergibt dies also ein Gleichungssystem mit $(n-r)\,r$ Gleichungen in den n Variablen $s_0, \ldots, s_{n-1}$ über L. Allerdings besteht dieses System aufgrund der vorkommenden $(s_{j-i})^{q^i}$ aus polynomialen Gleichungen von großem Grad. Werden die Gleichungen allerdings über K betrachtet, d.h. wird s_i ersetzt durch $(s_{i1}, \ldots, s_{in}) := \phi(s_i)$ für den Isomorphismus $\phi : L \to K^n$, so ergeben sich $n\,(n-r)\,r$ Gleichungen in den n^2 Variablen s_{ij} über K. Zudem sind diese Gleichungen linear, da die s_i nur in q-Potenzen in den vorherigen Gleichungen vorkamen. Dieses lineare Gleichungssystem ist also deutlich überbestimmt und liefert somit die Transformation s.

In [Cou99] analysiert Courtois diese Attacke und kommt für die oben schon erwähnte HFE-Challenge 1 zu Laufzeiten von 2^{152} Taktzyklen für die von Kipnis und Shamir vorgeschlagene Variante und auf eine Laufzeit von 2^{82} Taktzyklen für den eigenen Verbesserungsvorschlag mit Submatrizen. Beide Attacken liegen also außerhalb des als durchführbar anzusehenden Bereichs.

5.3 Zusammenfassung

Die in diesem Kapitel beschriebenen Attacken haben gemeinsam, dass ihre geschätzten Laufzeiten für übliche Parameterwahlen kaum besser bzw. meist schlechter als eine vollständige Suche sind. Nur die „Distillation Technique“ von Courtois ist etwas besser, aber

diese Attacke ist bisher weder ausführlich beschrieben, noch ist ihre Laufzeit gründlich untersucht. Denn aus den Beschreibungen in [Cou01a] ist zu vermuten, dass die dort geschätzte Komplexität dieser Attacke für $n = 80$ lediglich auf den Untersuchungen für $n = 21$ beruht, die auf den Fall $n = 80$ hochgerechnet wurden.
Bislang sind für HFE mit Parameterwahlen, die eine vollständige Suche ausschließen (d.h. im Wesentlichen $q^n > 2^{80}$, vgl. Abschnitt 2.3.4), also keine durchführbaren Attacken bekannt.
Courtois schließt aus seinen Beobachtungen in [Cou01a], dass ein einfaches HFE-System für $d > 128$ und $n > 80$ sicher ist.

Zudem sind alle hier vorgeschlagenen Attacken nur auf ein reines HFE-System anwendbar, d.h. auf ein System, das nicht durch die in Abschnitt 2.4 beschriebenen Perturbationen modifiziert ist. Für solche, modifizierten Systeme sind bislang keine Attacken bekannt, die an die Komplexität einer vollständigen Suche annähernd herankämen.

Anhang A

Inhalt der CD

Die beigelegte CD enthält folgende Dateien:

`HFE.pdf`	diese Arbeit im PDF-Format
`HFE.ps`	diese Arbeit im Postscript-Format
`HFEDaten.txt`	Ergebnisse der Simulationen, die mit der Prozedur `TestGB` durchgeführt wurden (vgl. Abschnitt 3.4.2)
`ZufDaten.txt`	Ergebnisse der Simulationen, die mit der Prozedur `TestGBZufall` durchgeführt wurden (vgl. Abschnitt 3.4.2)
`Singular\...`	die zur Durchführung der Simulationen benötigten Singular-Prozeduren

Die Dateien `HFEDaten.txt` und `ZufDaten.txt` liegen im ASCII-Format vor. In jeder Zeile dieser Dateien sind jeweils die Daten einer Simulation in folgenden Formaten enthalten:

In `HFEDaten.txt` enthält jede Zeile die Parameter q, n und d, die Anzahl der Lösungen des Systems und die zur Lösung benötigte Zeit in folgender Reihenfolge jeweils durch ein Leerzeichen getrennt:

```
q    n    d    Lösungsanzahl    benötigte Zeit
```

In `ZufDaten.txt` enthält jede Zeile die Parameter q und n, die Anzahl der Lösungen des Systems und die zur Lösung benötigte Zeit ebenfalls durch Leerzeichen getrennt in folgender Reihenfolge:

```
q    n    Lösungsanzahl    benötigte Zeit
```

Die Prozeduren im Verzeichnis `Singular` können in Singular entweder einzeln oder (nachdem sie in das Singular-Verzeichnis kopiert wurden) mit Hilfe der Datei `AlleProzeduren.sng` durch den Befehl `execute(read("AlleProzeduren.sng"));` eingebunden werden.

Literaturverzeichnis

[BCEMR94] B. Barkee, D. C. Can, J. Ecks, T. Moriarty, R. F. Ree: *Why You Cannot Even Hope to use Gröbner Bases in Public Key Cryptography: An Open Letter to a Scientist Who Failed and a Challenge to Those Who Have Not Yet Failed*, in: Journal of Symbolic Computation 18, 1994, S. 497-501

[BFS96] J. F. Buss, G. S. Frandsen, J. O. Shallit: *The computational complexity of some problems of linear algebra*, BRICS series report, RS-96-33, 1996, verfügbar unter `http://www.brics.dk/RS/96/33`

[BGMMVY92] I. F. Blake, X. Gao, A. J. Menezes, R. C. Mullin, S. A. Vanstone, T. Yaghoobian: *Applications of Finite Fields*, Kluwer Academic Publishers, 1992

[Bra79] G. Brassard: *A note on the complexity of cryptography*, IEEE Tran. Information Theory, Vol. IT-25, 1979, S. 232-233

[BS88] D. Bayer, M. Stillman: *On the complexity of computing syzygies*, in: Journal of Symbolic Computation 6, 1988, S. 135-147

[Bu65] B. Buchberger: *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Dissertation, Innsbruck, 1965

[Bu85] B. Buchberger: *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory*, in: N. K. Bose (Ed.): Multidimensional Systems Theory, D. Reidel Publishing Company, 1985, S. 184-232

[BW93] T. Becker, V. Weispfenning: *Gröbner Bases - A Computational Approach to Commutative Algebra*, Springer-Verlag, 1993

[CFS01] N. Courtois, M. Finiasz, N. Sendrier: *How to achieve a McEliece-based Digital Signature Scheme*, Cryptology ePrint Archive, Report 2001/010, 2001, verfügbar unter `http://eprint.iacr.org/`

[CGH88] L. Caniglia, A. Galligo, J. Heintz: *Some new effectivity bounds in computational geometry*, in: T. Mora (Ed.): Applied Algebra, Algebraic Al-

gorithms and Error-Correcting Codes, LNCS 357, Springer-Verlag, 1988, S. 131-152

[CGP01] N. Courtois, L. Goubin, J. Patarin: *QUARTZ, 128-Bit Long Digital Signatures*, in: D. Naccache (Ed.): Topics in Cryptology - CT-RSA 2001, LNCS 2020, Springer-Verlag, 2001

[CGP98a] N. Courtois, L. Goubin, J. Patarin: *Improved Algorithms for Isomorphisms of Polynomials*, in: K. Nyberg (Ed.) : Advances in Cryptology - EUROCRYPT '98, LNCS 1403, Springer-Verlag, 1998, (auch in [CGP99])

[CGP98b] N. Courtois, L. Goubin, J. Patarin: *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, in: K. Ohta, D. Pei (Eds.): Advances in Cryptology - Asiacrypt 1998, LNCS 1514, Springer-Verlag, 1998, (auch in [CGP99])

[CGP99] N. Courtois, L. Goubin, J. Patarin et al.: *Asymmetric Cryptography with Multivariate Polynomials over a Small Finite Field*, erhältlich unter `J.Patarin@frlv.bull.fr`

[CKPS00] N. Courtois, A. Klimov, J. Patarin, A. Shamir: *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, in: B. Preneel (Ed.): Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, Springer-Verlag, 2000

[CLO92] D. Cox, J. Little, D. O'Shea: *Ideals, Varieties, and Algorithms*, Springer-Verlag, 1992

[Coh99] A. M. Cohen: *Gröbner Bases, an Introduction*, in: A. M. Cohen, H. Cuypers, H. Sterk (Eds.): Some Tapas of Computer Algebra, Springer-Verlag, 1999, S. 1-33

[Cou01a] N. Courtois: *The Security of Hidden Field Equations (HFE)*, in: D. Naccache (Ed.): Topics in Cryptology - CT-RSA 2001, LNCS 2020, Springer-Verlag, 2001

[Cou01b] N. Courtois: *The HFE cryptosystem home page*, verfügbar unter `http://hfe.minrank.org`

[Cou01c] N. Courtois: *The security of cryptographic primitives based on multivariate algebraic problems: MQ, MinRank, IP, HFE*, Dissertation, noch nicht erschienen

[Cou99] N. Courtois: *The Shamir-Kipnis Attack on HFE and C^**, in [CGP99]

[Cox98] D. A. Cox: *Introduction to Gröbner Bases*, in: Proceedings of Symposia in Applied Mathematics, Vol. 53, 1998, S. 1-24

[Dub89] T. Dubé: *Quantitative analysis problems in computer algebra: Gröbner bases and the Nullstellensatz*, PhD Thesis, Courant Institute, New York University, New York, 1989

[Fel01] P. Felke: *Multivariate Kryptosysteme, insbesondere das Schema von Imai-Matsumoto*, Diplomarbeit, Universität Dortmund, 2001

[FGLM93] J. C. Faugère, P. Gianni, D. Lazard, T. Mora: *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering*, in: Journal of Symbolic Computation 16, 1993, S. 329-344

[GG99] J. von zur Gathen, J. Gerhard: *Modern Computer Algebra*, Cambridge University Press, 1999

[GJ79] M. R. Garey, D. S. Johnson: *Computers and intractability - a guide to the theory of NP-completeness*, Freeman, 1979

[GKP99] L. Goubin, A. Kipnis, J. Patarin: *Unbalanced Oil and Vinegar Signature Schemes*, in: J. Stern (Ed.): Advances in Cryptology - EUROCRYPT '99, LNCS 1592, Springer-Verlag, 1999, (auch in [CGP99])

[GP97a] L. Goubin, J. Patarin: *Asymmetric Cryptography with S-Boxes*, in: Y. Han, T. Okamoto, S. Quing (Eds.): Information and Communications Security, LNCS 1334, Springer-Verlag, 1997, S. 369-380, (auch in [CGP99])

[GP97b] L. Goubin, J. Patarin: *Trapdoor One-way Permutations and Multivariate Polynomials*, in: Y. Han, T. Okamoto, S. Quing (Eds.): Information and Communications Security, LNCS 1334, Springer-Verlag, 1997, S. 356-368, (auch in [CGP99])

[GPS01] G.-M. Greuel, G. Pfister, H. Schönemann: SINGULAR *2.0. A Computer Algebra System for Polynomial Computations*, Centre for Computer Algebra, University of Kaiserslautern, 2001, verfügbar unter `http://www.singular.uni-kl.de`

[GS92] J. von zur Gathen, V. Shoup: *Computing Frobenius maps and factoring polynomials*, in: Proc. 24th Annual ACM Symp. Theory of Computing, ACM Press, 1992

[Huy86] D. T. Huynh: *A superexponential lower bound for Gröbner bases and Church Rosser commutative Thue systems*, in: Information and Control 68, 1986, S. 196-206

[IM88] H. Imai, T. Matsumoto: *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption*, in: C. G. Günther (Ed.): Advances in Cryptology - EUROCRYPT '88, LNCS 330, Springer-Verlag, 1988, S. 419-453

[KS99] A. Kipnis, A. Shamir: *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, in: M. Wiener (Ed.): Advances in Cryptology - CRYPTO '99, LNCS 1666, Springer-Verlag, 1999, (auch in [CGP99])

[Laz83] D. Lazard: *Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations*, in: Proceedings EUROCAL 83, LNCS 162, Springer, 1983, S. 146-156

[Laz92] D. Lazard: *Solving Zero-dimensional Algebraic Systems*, in: Journal of Symbolic Computation 13, 1992, S. 117-131

[LN83] R. Lidl, H. Niederreiter: *Finite Fields*, Encyclopedia of Mathematics and its applications 20, Section: Algebra, Addison-Wesley, 1983

[May97] E. W. Mayr: *Some Complexity Results for Polynomial Ideals*, in: Journal of Complexity 13, 1997, S. 303-325

[MM82] E. W. Mayr, A. Meyer: *The complexity of the word problems for commutative semigroups and polynomial ideals*, in: Adv. in Math. 46(3), 1982, S. 305-329

[MM84] H. M. Möller, T. Mora: *Upper and lower bounds for the degree of Groebner bases*, in: J. Fitch (Ed.): EUROSAM 1984, LNCS 174, Springer-Verlag, 1984, S. 172-183

[Moh01] T. Moh: *On The Method of "XL" And Its Inefficiency to TTM*, Cryptology ePrint Archive, Report 2001/047, 2001, verfügbar unter `http://eprint.iacr.org/`

[Pat95] J. Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, in: D. Coppersmith (Ed.): Advances in Cryptology - CRYPTO '95, LNCS 963, Springer-Verlag, 1995, S. 248-261, (auch in [CGP99])

[Pat96a] J. Patarin: *Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*, in: U. M. Maurer (Ed.): Advances in Cryptology - EUROCRYPT '96, LNCS 1070, Springer Verlag, 1996, S. 33-48, (auch in [CGP99])

[Pat96b] J. Patarin: *Asymmetric Cryptography with a Hidden Monomial*, in: N. Koblitz (Ed.): Advances in Cryptology - CRYPTO '96, LNCS 1109, Springer-Verlag, 1996, S. 45-60, (auch in [CGP99])

[Weg99] I. Wegener: *Theoretische Informatik - eine algorithmische Einführung*, 2., durchges. Auflage, Teubner, 1999

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe.

Dortmund, im August 2001

Magnus Daum