Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik
Fachgebiet Theoretische Informatik

Public-Key Kryptographie mit multivariaten quadratischen Gleichungen

Julian Bilsky
Matrikelnummer: 10004858

Prüfer: Prof. Dr. rer. nat. Heribert Vollmer
Zweitprüfer: Dr. rer. nat. Arne Meier
Betreuer: M. Sc. Timon Barlag

14.04.2019

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und dabei nur die angegebeben Quellen und Hilfsmittel verwendet habe.

Hannover, 28.03.2020

Julian Bilsky

Inhaltsverzeichnis

1 Einleitung

Kryptographie ist nicht mehr aus dem täglichen Leben wegzudenken. In einem Zeitalter, in dem ein Großteil der Kommunikation über ein öffentliches Medium, wie das Internet stattfindet, ist es wichtig diese vor Angriffen zu schützen. Viele vertrauliche Daten, wie Bankinformationen, Passwörter und e-Mails werden verschlüsselt, damit Dritte keinen Zugriff darauf haben. Nur durch kryptographische Verfahren kann dieser Informationsaustausch gewährleistet werden.

Die Standards heutiger Verschlüsselungssysteme basieren auf verschiedenen mathematischen Problemen, wie der Primzahlfaktorisierung (RSA)[1], diskreten Logarithmen (DSA)[2] oder elliptischen Kurven (ECDSA)[3]. Diese sind mathematische Einwegfunktionen, die in eine Richtung leicht zu berechnen sind. Um sie zu invertieren, ist allerdings ein großer Berechnungsaufwand erforderlich, der Angreifer davon abhält die Nachricht zu entschlüsseln. Deswegen gelten diese Verfahren als sicher, da sie mit herkömmlichen Mitteln exponentiell viel Zeit benötigen, um sie ohne Schlüssel zu entziffern.

Doch je schneller Computer und Algorithmen werden, desto leichter wird es, Kryptographiesysteme zu brechen. Es findet ein ständiges Wettrüsten zwischen Angriffen auf Systeme und Optimierungen von Sicherheitskonzepten statt. Der nächste große Schritt werden voraussichtlich die Quantencomputer sein. Diese versprechen durch die Physik der Quantenmechanik und durch Algorithmen der Quanteninformatik schnellere Laufzeiten, als herkömmliche Rechner. Dies betrifft die Sicherheit der genannten Verschlüsselungssysteme. Denn nach heutigem Wissen werden diese Systeme durch Quantenrechner nicht mehr nutzbar sein. Quantenalgorithmen, wie der Shor-Algorithmus, sollen Probleme, wie das Faktorisieren von großen Zahlen in polynomieller Zeit berechnen können, was die Sicherheit dieser Konzepte zunichte macht.

Es muss jedoch beachtet werden, dass ausreichend große Quantencomputer noch nicht existieren und wir bisher nur die Theorie hinter diesen Verfahren betrachten können. Führende Firmen wie IBM und Google arbeiten bereits an Quantencomputern. Deren Leistung ist noch weit entfernt Kryptographiesysteme zu brechen.

Trotzdem sollten in naher Zukunft Alternativen eingeführt werden, damit weiterhin funktionierende, aber auch effiziente Sicherheitssysteme genutzt werden können. Es kann sich nicht darauf verlassen werden, dass klassische Verfahren sicher bleiben.

Eine dieser Alternativen, die bereits erforscht werden, ist die Multivariate-Quadratic-Equations Kryptographie. Diese wurde bereits 1988 eingeführt und basiert auf multivariaten Polynomen über endlichen Körpern. Bisher ist kein klassischer oder Quantum-

Algorithmus bekannt, der eine Lösung für ein System aus zufällig gewählten, quadratischen Gleichungen in polynomieller Zeit löst, weshalb derzeit davon ausgegangen wird, dass dieses Verfahren sicher bleibt.

2 Grundlagen

Zunächst betrachten wir die Grundlagen der Kryptographie, die für ein Public-Key Verfahren und digitale Signaturen von Bedeutung sind. Diese beiden Verfahren sind der hauptsächliche Verwendungszweck von multivariaten quadratischen ($\mathcal{MQ}$) Kryptosystemen.

Danach widmen wir uns endlichen Körpern, Erweiterungskörper über diesen und affinen Abbildungen, welche eine große Rolle in dem Kryptosystem spielen.

2.1 Public-Key Kryptographie

Das grundlegende Problem ist, wie zwei Parteien sicher miteinander kommunizieren können. Nachrichten müssen verschlüsselt werden, damit sie nicht gelesen werden können, sollten sie in die falschen Hände gelangen.

Es gibt symmetrische und asymmetrische Verschlüsselungsverfahren. Beim symmetrischen Verfahren benutzen beide Parteien denselben Schlüsseln zum Ver- und Entschlüsseln. Die Schwierigkeit liegt dabei beim Austausch des Schlüssels. Wenn in einem Kommunikationsnetz mit n Teilnehmern alle eine verschlüsselte Verbindung aufbauen wollen, müssen $n(n-1)/2$ Schlüsselaustausche stattfinden. Da bei Milliarden von Internet-Nutzern der organisatorische Aufwand viel zu groß wäre, wurde das asymmetrische Public-Key Verfahren eingeführt, mit welchem wir uns hauptsächlich beschäftigen werden.

Jede Partei besitzt einen öffentlichen und einen privaten Schlüssel. Der öffentliche Schlüssel ist für alle verfügbar und dient zum Verschlüsseln von Nachrichten. Jeder kann ihn benutzen und Nachrichten verschlüsseln, aber nur mit dem zugehörigen privaten Schlüssel können diese entschlüsselt werden. Die Idee für so ein Verfahren wurde 1976 von Diffie und Hellman eingeführt[4]. Das wohl bekannteste und am weitesten verbreitetste Public-Key Verfahren ist RSA, welches nach seinen Erfindern Rivest, Shamir und Adleman benannt ist.[1]

Es folgt eine formale Definition für solche Kryptosysteme.

Definition 1 ([5])**.** Ein Public-Key-Verschlüsselungsverfahren oder Public-Key-Kryptosystem ist ein Tupel (**K**, **P**, **C**, **KeyGen**, **Enc**, **Dec**) mit folgenden Eigenschaften:

1. **K** ist eine Menge von Paaren. Sie heißt Schlüsselraum. Ihre Elemente (e, d) heißen Schlüsselpaare. Ist $(e, d) \in \mathbf{K}$, so heißt die Komponente e öffentlicher Schlüssel und die Komponente d privater Schlüssel.

2. **P** ist eine Menge. Sie heißt Klartextraum. Ihre Elemente heißen Klartexte.

3. **C** ist eine Menge. Sie heißt Chiffretextraum. Ihre Elemente heißen Chiffretexte oder Schlüsseltexte.

4. **KeyGen** ist ein probabilistischer Polynomzeit-Algorithmus. Er heißt Schlüsselerzeugungsalgorithmus. Bei Eingabe von 1^k für ein $k \in \mathbb{N}$ gibt er ein Schlüsselpaar $(e, d) \in \mathbf{K}$ zurück. Wir schreiben dann $(e, d) \leftarrow \mathbf{KeyGen}(1^k)$. Mit $\mathbf{K}(k)$ bezeichnen wir die Menge aller Schlüsselpaare, die **KeyGen** bei Eingabe von 1^k zurückgeben kann. Mit $\mathbf{Pub}(k)$ bezeichnen wir die Menge aller öffentlichen Schlüssel, die als erste Komponente eines Schlüsselpaares $(e, d) \in \mathbf{K}(k)$ auftreten kann. Die Menge aller zweiten Komponenten dieser Schlüsselpaare bezeichnen wir mit $\mathbf{Priv}(k)$. Außerdem bezeichnet $\mathbf{P}(e) \subset \mathbf{P}$ die Menge der Klartexte, die mit einem öffentlichen Schlüssel e verschlüsselt werden können.

5. **Enc** ist ein probabilistischer Polynomzeit-Algorithmus. Er heißt Verschlüsselungsalgorithmus. Bei Eingabe von $1^k, k \in \mathbb{N}$, eines öffentlichen Schlüssels $e \in \mathbf{Pub}(k)$ und eines Klartextes $P \in \mathbf{P}(e)$ gibt er einen Chiffretext **C** zurück. Wir schreiben dann $\mathbf{C} \leftarrow \mathbf{Enc}(1^k, e, P)$.

6. **Dec** ist ein deterministischer Algorithmus. Er heißt Entschlüsselungsalgorithmus. Er entschlüsselt korrekt: Sei $k \in \mathbb{N}, (e, d) \in \mathbf{K}(k), \mathbf{P} \in \mathbf{P}(e)$ und $C \leftarrow \mathbf{Enc}(1^k, e, P)$. Dann gilt $\mathbf{P} \leftarrow \mathbf{Dec}(1^k, d, C)$.

Für ein Public-Key Kryptosystem brauchen wir außerdem zwei weitere Dinge: ein schwer zu berechnendes mathematisches Problem, welches den Angreifer davon abhält unseren verschlüsselten Text zu entschlüsseln und eine mathematische Falltür, die uns ermöglicht, mithilfe des privaten Schlüssels die Nachricht zu lesen.

Bei RSA ist das schwierige Problem das Faktorisieren von Primzahlen. Bei $\mathcal{MQ}$-Systemen nutzen wir, dass das Lösen von multivariaten quadratischen Gleichungssystemen NP-vollständig ist.

Wie das Public-Key Verfahren genau durch multivariate quadratische Gleichungen realisiert wird, sehen wir in den folgenden Abschnitten.

2.2 Digitale Signaturen

Ein weiterer Anwendungsbereich von $\mathcal{MQ}$-Systemen sind digitale Signaturen. Diese werden verwendet, um Authentizität und Integrität von Dokumenten oder Software zu beweisen. Mit solchen Signaturen kann also der Erzeuger eines Dokumentes verifiziert und bestätigt werden, dass das Nachricht nicht verfälscht wurde.

Ein digitales Signaturverfahren besteht aus einem Schlüsselerzeugungsalgorithmus, einem Signieralgorithmus und einem Verifikationsalgorithmus. Durch den Schlüsselerzeugungsalgorithmus werden Schlüsselpaare (e, d) erzeugt. d ist der private Signierschlüssel und e der öffentliche Verifikationsschlüssel. Der Signieralgorithmus berechnet aus einem Dokument x und dem Schlüssel d eine digitale Signatur s. Der Verifikationsalgorithmus bekommt als Eingabe das Dokument x, die digitale Signatur s und den öffentliche Schlüssel e. Wenn s aus dem zu e gehörenden Schlüssel d aus x berechnet wurde, gibt der Algorithmus „gültig“ zurück und ansonsten „ungültig“[5].

Wir werden sehen, dass Verschlüsselungsverfahren und digitale Signaturen stark aufeinander aufbauen. Bei beiden Methoden geht es darum, den privaten Schlüssel zu invertieren.

2.3 Endliche Körper

Endliche Körper bestehen aus einer endlichen Menge von Elementen und zwei Operationen: Addition und Multiplikation. Werden zwei Elemente der Menge durch eine Operation verknüpft, muss das Ergebnis wieder im Körper liegen. Eine formale Definition sieht wie folgt aus:

Definition 2 ([6])**.** Sei $\mathbb{F}$ eine Menge von $q \in \mathbb{N}$ Elementen mit den zwei Operatoren Addition $+\colon \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ und Multiplikation $\cdot\colon \mathbb{F} \times \mathbb{F} \to \mathbb{F}$. Wir nennen $(\mathbb{F}, +, \cdot)$ einen Körper, wenn die folgenden Axiome erfüllt sind:

1. Additive abelsche Gruppe$(\mathbb{F}, +)$:
   (a) Assoziativität: $\forall a, b, c \in \mathbb{F} : (a + (b + c)) = ((a + b) + c)$
   (b) Kommutativität: $\forall a, b \in \mathbb{F} : a + b = b + a$
   (c) Neutrales Element: $\exists e \in \mathbb{F} : \forall a \in \mathbb{F} : a + e = a$

(d) Additives Inverses: $\forall a \in \mathbb{F}\, \exists a' \in \mathbb{F} : a + a' = 0$

2. Multiplikative abelsche Gruppe $(\mathbb{F}^*, \cdot)$ für $\mathbb{F}^* := \mathbb{F} \setminus \{0\}$

   (a) Assoziativität: $\forall a, b, c \in \mathbb{F} : ((a \cdot b) \cdot c) = (a \cdot (b \cdot c))$

   (b) Kommutativität: $\forall a, b \in \mathbb{F} : a \cdot b = b \cdot a$

   (c) Neutrales Element: $\exists e \in \mathbb{F} : \forall a \in \mathbb{F} a \cdot e = a$

   (d) Multiplikatives Inverses: $\forall a \in \mathbb{F}^* \exists a' \in \mathbb{F}^* : a \cdot a' = 1$

3. Distributivgesetz: $\forall a, b, c \in \mathbb{F} : a \cdot (b + c) = a \cdot b + a \cdot c$

Definition 3. Sei q eine Primzahl, $\mathbb{F} \coloneqq \{0, ..., q-1\}$ und Addition und Multiplikation normale Ganzzahladdition und Multiplikation modulo der Primzahl q. Dann nennen wir $(\mathbb{F}, +, \cdot)$ einen Primkörper. Im Folgenden schreiben wir auch $\mathbb{F}$ bzw. $\mathbb{F}_q$, wenn klar ist, dass ein Körper gemeint ist.

Beispiel 4. Der Körper den wir hauptsächlich betrachten werden ist $\mathbb{F}_2$. Dieser wird auch GF(2) genannt, was für Galois Field steht. Da der Körper nur zwei Elemente $\{0, 1\}$ hat, ist die gesamte Addition und Multiplikation in zwei Tabellen darstellbar:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

In der folgenden Definition sprechen wir von Ringen und irreduziblen Polynomen. Ein Ring ist lose gesagt, eine Verallgemeinerung von Körpern, dessen Elemente keine Inversen besitzen müssen[7]. Die Division von Elementen ist in Ringen dementsprechend nicht gefordert. Ein irreduzibles Polynom ist ein Polynom, welches nicht in kleinere, nicht konstante Polynome faktorisiert werden kann.

Definition 5 ([8])**.** Sei $\mathbb{F}$ ein Körper. Ein nicht konstantes Polynom $f(x)$ ist irreduzibel über $\mathbb{F}$ wenn es keine Polynome $g(x)$ und $h(x)$ gibt, sodass $f(x)$ das Produkt von $g(x), h(x)$ ist und Grad von $g(x)$ und der Grad von $h(x)$ beide kleiner als der von $f(x)$ sind.

Definition 6 ([7])**.** Sei $\mathbb{F}$ ein Körper und $i(t) \in \mathbb{F}[t]$ ein irreduzibles Polynom vom Grad n aus dem Polynomring $\mathbb{F}[t]$. Sei $\mathbb{E} \coloneqq \mathbb{F}[t]/i(t)$ die Äquivalenzklasse der Polynome modulo $i(t)$. Dann bildet diese mit den Operationen „+“, der Addition von Polynomen und „$\cdot$“, der Multiplikation von Polynomen modulo $i(t)$ den Körper $(\mathbb{E}, +, \cdot)$.

Diese Definition erlaubt es uns, Erweiterungskörper der Form $\mathbb{F}_{p^n}$ zu bilden. Damit sind alle möglichen endlichen Körper abgedeckt, da die Anzahl der Elemente immer eine Primzahlpotenz ist. [7]

Beispiel 7. Um den Körper $\mathbb{F}_{2^2}$ zu bilden, brauchen wir ein irreduzibles Polynom $g(X)$ vom Grad 2 in $\mathbb{F}_2[X]$. Das einzige existierende ist $g(X) = X^2 + X + 1$. Die Elemente des Körpers $\mathbb{F}_{2^2}$ sind die Reste die bei der Division durch $g(X)$ entstehen können, also alle Polynome vom Grad kleiner 2.

$$\begin{aligned}\mathbb{F}_{2^2} &= \mathbb{F}_2[X]/(X^2 + X + 1) \\ &= \{0, 1, X, X + 1\}\end{aligned}$$

Die Operationen sind die übliche Addition von Polynomen und Multiplikation modulo $g(X) = X^2 + X + 1$. Beide werden durch folgende Matrizen vollständig beschrieben

$+$	0	1	X	$X+1$
0	0	1	X	$X+1$
1	1	0	$X+1$	X
X	X	$X+1$	0	1
$X+1$	$X+1$	X	1	0

$\times$	0	1	X	$X+1$
0	0	0	0	0
1	0	1	X	$X+1$
X	0	X	$X+1$	1
$X+1$	0	$X+1$	1	X

Definition 8 (Frobeniushomomorphismus)**.** Sei $\mathbb{F}$ ein endlicher Körper und $q \coloneqq |\mathbb{F}|$ die Anzahl der Elemente und damit eine Primzahl. Dann gilt $\forall x \in \mathbb{F} : x^q = x$.

2.4 Körpererweiterung

Da manche Verfahren nicht nur auf einem Körper definiert sind, sondern auch einen Erweiterungskörper nutzen, führen wir auch diese ein.

Definition 9. Eine Körpererweiterung besteht aus zwei Körpern K und L mit $K \subset L$. Sei K eine Teilmenge von L, die neutralen Elemente der Addition und Multiplikation enthält und bezüglich der Verknüpfungen über L abgeschlossen ist, d.h. wenn

$$a, b, c, 0, 1 \in K, c \neq 0 \rightarrow a - b, ab, c^{-1} \in K$$

gilt, dann nennen wir K einen Teilkörper von L und L einen Erweiterungskörper von K.

Der Erweiterungskörper L ist außerdem ein Vektorraum über K. Der Grad einer Körpererweiterung $K \subset L$ ist die Vektorraumdimension[9].

Beispiel 10. Ein klassisches Beispiel für Körpererweiterungen sind die rationalen Zahlen $\mathbb{Q}$, die reellen Zahlen $\mathbb{R}$ und die komplexen Zahlen $\mathbb{C}$. Die reellen Zahlen sind ein Erweiterungskörper der rationalen Zahlen und ein Teilkörper der komplexen Zahlen. In unserem Beispiel 7 ist $\mathbb{F}_2$ der Teilkörper des Erweiterungskörpers $\mathbb{F}_{2^2}$.

Dass der Erweiterungskörper als Vektorraum angesehen werden kann, sehen wir durch einen Isomorphismus zwischen dem Erweiterungskörper $\mathbb{E}$ und dem Vektorraum $\mathbb{F}^n$. Alle Elemente $a \in \mathbb{E}$ haben die Form:

$$a_{n-1}t^{n-1} + ... + a_1 t + a_0 \,\text{mit}\, a_i \in \mathbb{F}$$

Außerdem kann ein Vektor $b \in \mathbb{F}^n$ dargestellt werden als $(b_0, ..., b_{n-1})$ mit $b_i \in \mathbb{F}$.

Definition 11 ([6])**.** Sei $\mathbb{E}$ der Erweiterungskörper vom Grad n des Teilkörpers $\mathbb{F}$ und $\mathbb{F}^n$ der dazugehörige Vektorraum. Dann ist $\phi\colon \mathbb{E} \to \mathbb{F}^n$

$$\phi(a) \coloneqq b \,\text{und}\, b_i \coloneqq a_i \,\text{für}\, 0 \leq i < n$$

für $a_0, ..., a_{n-1}, b_0, ..., b_{n-1} \in \mathbb{F}$ die Bijektion zwischen $\mathbb{E}$ und $\mathbb{F}^n$. Die inverse Funktion ϕ^{-1} ist definiert als $\phi(\phi^{-1}(b)) = b$ für alle $b \in \mathbb{F}^n$ und $\phi^{-1}(\phi(a)) = a$ für alle $a \in \mathbb{E}$.

Diese Definition wird besonders nützlich, wenn wir $\mathcal{MQ}$-Systeme über Teilkörpern und Körpererweiterungen betrachten.

2.5 Affine Abbildungen

Genau wie endliche Körper spielen affine Abbildungen eine wichtige Rolle bei $\mathcal{MQ}$-Kryptosystemen. Sie sind Abbildungen zwischen affinen Räumen, bei der Kollinearität, Parallelität und Teilverhältnisse erhalten bleiben[10]. In $\mathcal{MQ}$-Systemen werden die privaten Polynomgleichungen, durch Komposition mit zwei affinen Abbildungen versteckt.

Für die Definition einer affinen Abbildung benötigen wir zunächst die Definition einer linearen Abbildung und eines affinen Raumes.

Definition 12. Sei A eine Menge, K ein Körper und V ein K-Vektorraum. Das Paar (A, V) heißt affiner Raum, wenn eine Operation $+ : A \times V \to A$ existiert, die dem Paar (P, v) mit $P \in A, v \in V$ das Element $P + v$ zuordnet, sodass gilt:

$$(P + v) + w = P + (v + w) \,\forall\, P \in A, v, w \in V$$
$$\forall\, P, Q \in A \,\exists\, v \in V : Q = P + v$$

v ist der Verbindungsvektor von P nach Q und wird mit $\overrightarrow{PQ}$ bezeichnet.

Definition 13 ([10])**.** Seien K ein Körper und V, W Vektorräume über K. Eine Abbildung $f : V \to W$ ist linear, falls für alle $v, v' \in V$ und alle $k \in K$ gilt:

$$f(v + v') = f(v) + f(v') \,(\text{Additivität})$$
$$f(k \cdot v) = k \cdot f(v) \,(\text{Homogenität})$$

Definition 14 ([11])**.** Seien (A, V_A), (B, V_B) affine Räume. Eine Abbildung $f : A \to B$ heißt affine Abbildung, wenn es eine lineare Abbildung $\varphi : V_A \to V_B$ zwischen den zugehörigen Vektorräumen gibt, so dass für alle Paare von Punkten $P, Q \in A$ gilt:

$$\overrightarrow{f(P)f(Q)} = \varphi(\overrightarrow{PQ})$$

Definition 15. Sei $S(x)$ eine affine Abbildung und $M_S \in \mathbb{F}^{n \times n}$ eine $(n \times n)$ Matrix und $v_s \in \mathbb{F}^n$ ein Vektor. Sei außerdem $S(x) := M_s x + v_s$. Dann nennen wir dies „*Matrix Darstellung*“ der affinen Abbildung S.

Definition 16. Seien $s_1, ..., s_n$ Polynome vom Grad 1 über $\mathbb{F}$, sodass $s_i(x_1, ..., x_n) := \beta_{i,1}x_1 + ... + \beta_{i,n}x_n + \alpha_i$ mit $1 \leq i, j \leq n$ und $\alpha_i, \beta_{i,j} \in \mathbb{F}$. Sei $S(x) := (s_1(x), ..., s_n(x))$ wobei $x := (x_1, ..., x_n)$ ein Vektor über $\mathbb{F}^n$ ist. Dann nennen wir $S(x)$ die „*multivariate Darstellung*“ der affinen Abbildung S.

Die Klasse der affinen Abbildungen $\mathbb{F}^n \to \mathbb{F}^n$ nennen wir $\mathit{Aff}(\mathbb{F}^n)$. Wir werden bei den Abbildungen von Bijektionen ausgehen, deswegen nennen wir die Klasse der invertierten affinen Abbildungen $\mathit{Aff}^{-1}(\mathbb{F}^n)$.

3 Multivariate Public-Key Kryptographie

3.1 $\mathcal{MQ}$-Konstruktion

Sei $n \in \mathbb{N}$ die Anzahl der Variablen, $m \in \mathbb{N}$ Die Anzahl der Gleichungen und $d \in \mathbb{N}$ der Grad des Systems. Die Variablen $x_1, ..., x_n$ sind Elemente aus $\mathbb{F}$. Der Vektor v mit Komponenten $v_1, ..., v_d \in \{0, ..., n\}$

$$\mathcal{V}_n^d := \begin{cases} \{0\}, & \text{für } d = 0 \\ \{v \in \{0, ..., n\}^d : i \leq j \Rightarrow v_i \leq v_j\}, & \text{sonst} \end{cases}$$

$\mathcal{V}_n^d$ ist nun die Menge aller sortierten Vektoren mit $d \in \mathbb{N}$ Elementen aus $\{0, ..., n\}$. Wenn wir über diese Menge iterieren, erhalten wir alle möglichen Kombinationen von Variablen von Grad d. Dadurch können wir nun die generellen multivariaten Polynomgleichungen definieren, wobei $\mathcal{P} := (p_1, ..., p_m)$ und alle p_i der Form sind:

$$p_i(x_1, ..., x_n) := \sum_{v \in \mathcal{V}_n^d} \gamma_{i,v} \prod x_{v_j} \; fr \; 1 \leq i \leq m$$

Alle Koeffizienten $\gamma_{i,v} \in \mathbb{F}$ und $v \in \mathcal{V}_n^d$. Bei einem Multivariaten Polynomgleichungssystem vom Grad 2, sprechen wir von Multivariaten Quadratischen Polynomen. Die Klasse der dazugehörigen polynomiellen Vektoren nennen wir $\mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$. Die Polynome haben dabei folgende Form:

$$
\begin{aligned}
p_1(x_1, ..., x_n) &\coloneqq \sum_{1 \le j \le k \le n} \gamma_{1,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{1,j} x_j + \alpha_1 \\
&\vdots \\
p_i(x_1, ..., x_n) &\coloneqq \sum_{1 \le j \le k \le n} \gamma_{i,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{i,j} x_j + \alpha_i \\
&\vdots \\
p_m(x_1, ..., x_n) &\coloneqq \sum_{1 \le j \le k \le n} \gamma_{m,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{m,j} x_j + \alpha_m
\end{aligned}
$$

Die Koeffizienten $\gamma_{i,j,k}, \beta_{i,j}, \alpha_i \in \mathbb{F}$ nennen wir quadratisch ($\gamma_{i,j,k}$), linear ($\beta_{i,j}$) und konstant (α_i). Über dem Körper $\mathbb{F}_2$ können wir $j \le k$ zu $j < k$ verallgemeinern, da nach Definition 8 $x^2 = x$ gilt.

Beispiel 17. Ein Polynomvektor könnte dann wie folgt aussehen:

$$
\begin{aligned}
y_1 &= x_1x_2 + x_1x_3 + x_2x_4 + x_2x_5 + x_4x_5 + x_2 + x_4 \\
y_2 &= x_1x_4 + x_2x_3 + x_4x_6 + x_1 + 1 \\
y_3 &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_3x_5 + x_4x_5 + x_1 \\
y_4 &= x_1x_2 + x_3x_5 + 1 \\
y_5 &= x_1x_3 + x_1x_4 + x_2x_3 + x_3x_5 + x_4x_5 + x_3 + x_4
\end{aligned}
$$

3.2 Falltüren

Wie eingangs erwähnt, brauchen wir für ein Public-Key Kryptosystem ein schwer zu berechnendes Problem und eine Falltür. Es gibt verschiedene Wege eine Falltür in ein $\mathcal{MQ}$-System einzubauen, doch meistens werden die privaten Gleichungen P' durch Komposition mit zwei affinen Abbildungen versteckt. Der Aufbau des Systems ist $P = T \circ P' \circ S : \mathbb{F}^n \to \mathbb{F}^m$, wobei $\mathbb{F}$ unser endlicher Körper ist. Unser öffentlicher Schlüssel ist hier P und unser privater Schlüssel besteht aus den affinen Abbildungen S, T und dem privaten Polynomvektor P'. Im Folgenden ist x unser Dokumentenvektor/Klartext und y die digitale Signatur/Chiffretext.

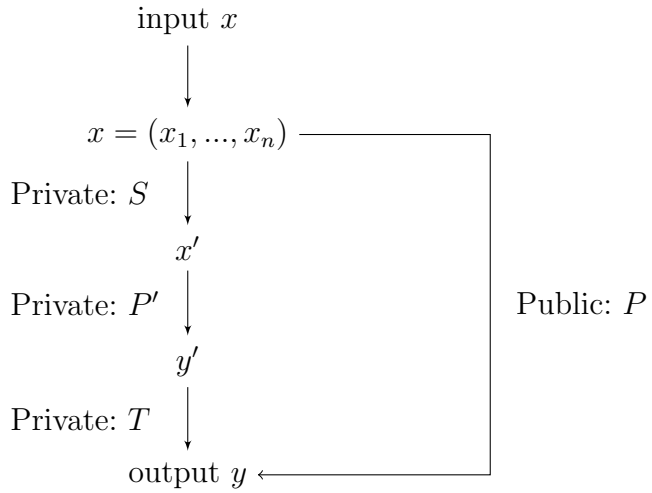

Abbildung 1: Darstellung einer MQ-Trapdoor

Die privaten Polynome P' gehört dabei zu einer speziellen Matrix, die leicht zu invertieren ist [12]. Der Aufbau des privaten Schlüssels ist hier $\mathit{Aff}^{-1}(\mathbb{F}^n, \mathbb{F}^n) \times \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m) \times \mathit{Aff}^{-1}(\mathbb{F}^m, \mathbb{F}^m)$. Die beiden affinen Abbildungen verstecken dabei die Struktur des Polynomvektors P' vor Angreifern. Für einen gegebenen Vektor $x \in \mathbb{F}^n$ ist die Lösung von $P(x)$ offensichtlich effizient berechenbar. Die umgekehrte Richtung, nämlich zu einem gegebenen $y \in \mathbb{F}^m$ ein passendes $x \in \mathbb{F}^n$ zu finden sodass gilt

$$\begin{aligned} p_1(x_1, ..., x_n) - y_1 &= 0 \\ &\vdots \\ p_i(x_1, ..., x_n) - y_i &= 0 \\ &\vdots \\ p_m(x_1, ..., x_n) - y_m &= 0 \end{aligned}$$

ist selbst über dem Körper $\mathbb{F}_2$ ein schwieriges mathematisches Problem[13].

3.3 Ver- und Entschlüsselung

$P(x) = y$ ist in der Regel nicht injektiv, also kann es mehrere $x \in \mathbb{F}^n$ geben für die die Gleichung erfüllt ist. Deswegen wird beim Verschlüsseln der Hash vom x Vektor gebildet, um beim Entschlüsseln den richtigen Eingabevektor eindeutig identifizieren zu können[6]. Zuerst wird also die Lösung y vom Polynomvektor P berechnet und danach der Hash vom Eingabevektor berechnet.

1. $y = P(x)$
2. $\tilde{x} = H(x)$

$H(x)$ ist eine beliebige kollisionsresistente Hashfunktion mit $H(x) : \mathbb{F}^n \to \{0,1\}^h$ wobei h die Länge des Hash-Strings ist. Die verschlüsselte Nachricht besteht nun aus dem Paar $(y, \tilde{x}) \in \mathbb{F}^m \times \{0,1\}^h$

Um die Nachricht zu entschlüsseln, wird nun der private Schlüssel verwendet. Zuerst müssen die Inversen der privaten Komponenten berechnet werden: S^{-1}, P'^{-1}, T^{-1}. Für einen Nachrichtenvektor y berechnen wir $y' \coloneqq T^{-1}(y)$, gefolgt von $x' \coloneqq P'^{-1}(y')$ und schließlich $x \coloneqq S^{-1}(x')$. S invertieren wir mithilfe der Matrixdarstellung. Die Funktion kann dargestellt werden als $S(x) \coloneqq Mx+v$. Somit ist ihre Inverse gegeben durch $S'(x) \coloneqq M^{-1}(x-v)$. T invertieren wir analog. Die Invertierung der Matrix P' unterscheidet sich für die einzelnen Falltüren. Die Vorgehen besprechen wir in den jeweiligen Sektionen.

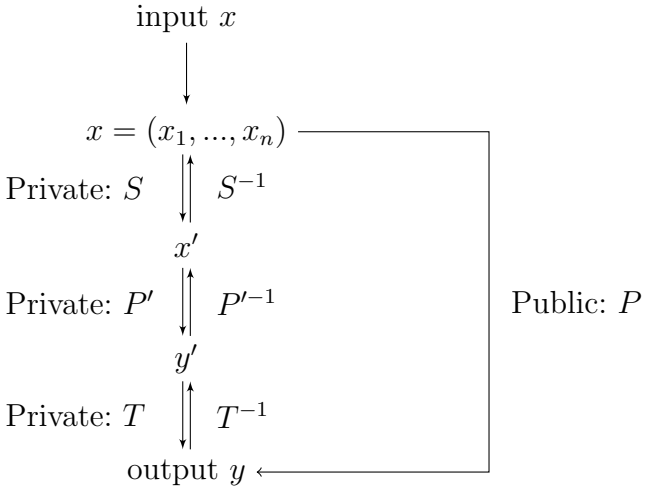

Abbildung 2: Darstellung einer MQ-Trapdoor mit inversen Funktionen

3.4 Digitale Signatur

Beim Erstellen der digitalen Signatur berechnen wir $x \in \mathbb{F}^n$ für den Dokumentenvektor $y \in \mathbb{F}^m$. Dafür müssen wir alle Schritte umkehren, also berechnen wir die Inversen T^{-1}, P'^{-1}, S^{-1} wie bei der Entschlüsselung. Die Signatur berechnen wir analog zum Entschlüsseln: $x \coloneqq S^{-1}(P'^{-1}(T^{-1}(y)))$.

Um die digitale Signatur zu überprüfen, berechnen wir einfach die Lösung der Polynome mit Eingabevektor $x \in \mathbb{F}^n$ und vergleichen mit dem Nachrichtenvektor $y \in \mathbb{F}^m$:

$$\begin{aligned} y_1 &\stackrel{?}{=} p_1(x_1, ..., x_n) \\ &\vdots \\ y_m &\stackrel{?}{=} p_m(x_1, ..., x_n) \end{aligned}$$

Stimmt die Lösung der Polynome mit dem gegebenen Vektor y überein, akzeptieren wir die Signatur, ansonsten lehnen wir sie ab.

Ob sich ein $\mathcal{MQ}$-Verfahren für Ver- und Entschlüsselung oder für digitale Signaturen eignet, hängt von der Injektivität bzw. Surjektivität der zentralen Polynome P' ab. Verschlüsselungsschemen setzten Injektivität voraus, um einen eindeutigen Klartext wiederherstellen zu können, wohingegen für digitale Signaturen ein surjektives Polynomsystem benötigt wird, damit jedes Dokument signiert werden kann[14].

4 Konstruktion von $\mathcal{MQ}$-Falltüren

Wir haben nun den generellen Aufbau eines $\mathcal{MQ}$-Systems betrachtet und wie eine Falltür darin versteckt wird. In diesem Abschnitt geht es um spezielle Konstruktionen der Falltüren von $\mathcal{MQ}$-Systemen. Die Hauptaufgabe der verschiedenen Falltüren ist es, die Inverse der geheimen Polynome P' zu berechnen, um Chiffretexte zu entschlüsseln oder Dokumente zu signieren.

Manche Verfahren benutzen nur einen einfachen endlichen Körper (UOV, STS), während andere ein Teil- und Erweiterungskörper verwenden (MIA, HFE).

4.1 Unbalanced Oil and Vinegar (UOV)

Das *Oil and Vinegar Signatur Sheme* wurde erstmal in 1997 von J. Patarin vorgestellt[15]. Die Idee lag darin sogenannte Oil- und Vinegar-Variablen miteinander zu vermischen. Wie bei Öl und Essig in einem Salatdressing werden die Variablen nicht komplett miteinander vermischt, woher der Name kommt. Bereits ein Jahr später wurde das Schema von A. Kipnis and A. Shamir gebrochen, woraufhin 1999 Unbalanced Oil and Vinegar vorgestellt wurde, welches eine Verallgemeinerung des Originalschemas ist. Dabei werden nicht Oil- und Vinegar-Variablen zu gleichen Anteilen verwendet, sondern immer mehr Vinegar als Oil-Variablen benutzt[16].

Für die Anteile der Variablen wird meistens $v \geq 2o$ empfohlen, wobei $v \in \mathbb{N}$ die Anzahl der Vinegar und $o \in \mathbb{N}$ die Anzahl der Oil-Variablen repräsentiert [17]. Da die Anzahl der Gleichungen der Anzahl der Oil-Variablen entsprechen muss und wir mehr Oil als Vinegar Unbekannte brauchen, ist das Schema surjektiv. Damit eignet sich das Verfahren nur für digitale Signaturen.

Die $n \in \mathbb{N}$ zentralen Gleichungen haben folgende Form:

$$y_i := \sum_{j=1}^{o}\sum_{k=1}^{v} \epsilon_{i,j,k} x_j \breve{x}_k + \sum_{j=1}^{v}\sum_{k=1}^{v} \delta_{i,j,k} \breve{x}_j \breve{x}_k + \sum_{j=1}^{o} \gamma_{i,j} x_j + \sum_{j=1}^{v} \beta_{i,j} \breve{x}_j + \alpha'_i \text{ für } 1 \leq i \leq n$$

Die Koeffizienten $\epsilon_{i,j,k}, \delta_{i,j,k}, \gamma_{i,j}, \beta_{i,j}, \alpha_i \in \mathbb{F}$ sind die geheimen Koeffizienten der n Gleichungen. Die Unbekannten $x_1, ..., x_o \in \mathbb{F}$ sind die Oil Variablen und $\breve{x}_1, ..., \breve{x}_v \in \mathbb{F}$ die Vinegar Variablen. Die Vinegar-Variablen tauchen dabei in quadratischen Termen auf, die Oil-Variablen werden hingegen nur mit den Vinegar Unbekannten multipliziert.

Wir brauchen für dieses Schema nur eine affine Abbildung $S : \mathbb{F}^{n+v} \to \mathbb{F}^{n+v}$, die zweite Abbildung T wird ausgelassen oder ist die Identität, da sie durch die Mischung der Variablen im Gleichungssystem P' enthalten ist. Unserer Zentrales Polynomsystem ist somit der Form $\mathbb{F}^{o+v} \to \mathbb{F}^n$

Um die privaten Gleichungen P' zu invertieren, nutzen wir, dass die Variablen nicht vollstängt vermischt sind. Für ein Urbild $x \in \mathbb{F}^{o+v}$ von $y \in \mathbb{F}^n$ werden die Werte der Vinegar-Variablen $\breve{x}_1, ..., \breve{x}_v$ zufällig gewählt und in die Polyome $y_1, ..., y_n$ eingesetzt. Dadurch erhalten wir ein Gleichungssystem mit o linearen Gleichungen in den o Oil-Variablen $x_1, ..., x_o$. Dieses Gleichungssystem können wir mit dem Gaußschem Eliminationsverfahren lösen. Wenn wir eine Lösung finden, haben wir das Urbild. Gibt es keine Lösung, wiederholen wir den Vorgang, bis eine Lösung gefunden wird [14].

Beispiel 18. Sei $\mathbb{F}_7$ unser endlicher Körper, $o = v = 2$ und unsere Gleichungen $P' = (p_1, p_2)$ ist gegeben durch

$$p_1(x_1, ..., x_4) = 4x_1^2 + 2x_1x_2 + 5x_1x_4 + 3x_2^2 + x_2x_4 + x_1 + 4x_2 + x_3 + 3x_4 + 3$$
$$p_2(x_1, ..., x_4) = 3x_1^2 + x_1x_2 + 6x_1x_3 + 2x_2x_3 + 2x_2x_4 + 3x_1 + x_2 + 2x_3 + x_4 + 4$$

Unsere Nachricht ist der Vektor $y = (3, 2)$ und um eine Signatur zu erstellen suchen wir ein Urbild $x = (x_1, ..., x_4)$ welches unsere Gleichungen erfüllt. Um das Urbild x von y zu finden, geben wir unseren Vinegar-Variablen x_1 und x_2 zufällige Werte, zum Beispiel $x_1 = 1$, $x_2 = 4$. Dann setzen wir sie in die Gleichungen $p_1 = y_1 = 3$ und $p_2 = y_2 = 2$ ein und erhalten

$$3 = x_3 \ + 5x_4 + 3$$
$$2 = 2x_3 + 2x_4 + 4$$

Durch das Lösen des linearen Gleichungssystems erhalten wir $x_3 = 4$, $x_4 = 2$. Das gesuchte Urbild von $y = (3, 2) \in \mathbb{F}_7^2$ und somit unsere digitale Signatur ist also $x = (1, 4, 4, 2) \in \mathbb{F}_7^4$.

4.2 Step-wise Triangular Sheme (STS)

In dem *Step-wise Triangular Sheme* wird das private Polynomsystem „Schritt für Schritt“ aufgebaut. Der Parameter r bestimmt die Anzahl der Gleichungen sowie die Anzahl der Variablen die bei jedem Schritt hinzukommen.

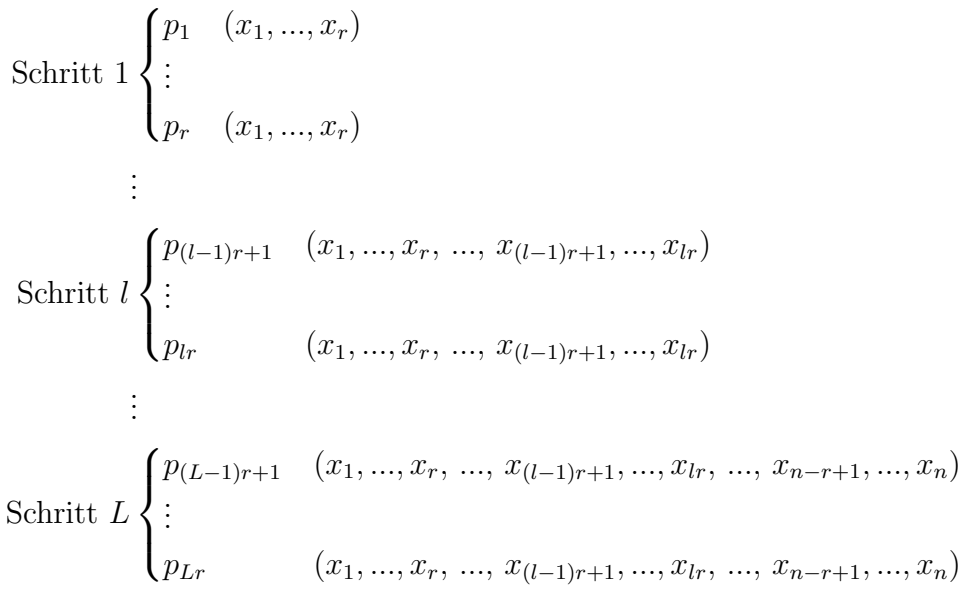

Abbildung 3: STS mit gleichbleibender Schrittgröße

Wie üblich ist m die Anzahl der Gleichungen und n die Anzahl der Variablen. Hinzu kommen L, die Anzahl der Schritte und r, die Schrittgröße. Sei nun $r_1, ..., r_L \in \mathbb{N}$ sodass $r_1 + ... + r_L = n$ und $m_1, ..., m_L \in \mathbb{N}$ sodass $m_1 + ... + m_L = m$. r_l ist die Anzahl der Variablen die pro Schritt hinzukommen und m_l die Anzahl der Gleichungen[6].

Bei der Abbildung 3 handelt es sich um „regular STS“. Hierbei gilt $r_1 = ... = r_L = m_1 = ...m_L$.

Für jeden neuen Schritt kommen bei regular STS r Variablen hinzu. Um das private Polynomsystem P' zu invertieren und eine Nachricht zu entschlüsseln, brauch der legitime Nutzer auf jeder Stufe höchsten q^r Versuche um für ein gegebenes y geeignete x-Werte zu finden. Der Arbeitsaufwand zum Entschlüsseln wächst also exponentiell mit r, weshalb dieser Wert klein gehalten werden muss, für eine praktische Anwendung mit diesem Verfahren[18]. Da STS in der Regel keine Bijektion bildet, ist es nötig wie in Sektion 3.3 erwähnt, einen Hash als Redundanz hinzuzunehmen.

Beispiel 19. Sei unser Körper $\mathbb{F}_2$, $r = m = 2$. Die Nachricht die wir entschlüsseln wollen

ist $y = (1, 0, 0, 1)$ unserer Polynomsystem ist gegeben durch

$$\begin{aligned} y_1 &= x_1x_2 + x_1 + x_2 \\ y_2 &= x_1x_2 + x_1 \\ y_3 &= x_1x_2 + x_1x_3 + x_2x_4 + x_2 + x_3 \\ y_4 &= x_1x_3 + x_2x_3 + x_3x_4 + x_1 + x_3 + 1 \end{aligned}$$

Zu beachten ist, dass die Koeffizienten $\gamma_{i,j,k}, \beta_{i,j}, \alpha_i$ der quadratischen, linearen und konstanten Termen zufällig gewählt werden.
Für jeden Schritt haben wir nun $p^r = 2^2$ Möglichkeiten die Variablen zu verteilen. Für den ersten Schritt probieren wir alle Möglichkeiten aus, bis wir sehen, dass die Zuweisung $x_1 = 1, x_2 = 1$ die Gleichungen erfüllt.

$$\begin{aligned} y_1 &= 1 = 1 \times 1 + 1 + 1 \\ y_2 &= 0 = 1 \times 1 + 1 \end{aligned}$$

Für den nächsten Schritt gibt es wieder nur vier Möglichkeiten, da wir die ersten zwei Variablen schon kennen und diese in die Gleichungen einsetzen können.

$$\begin{aligned} y_3 &= 0 = 1 + x_3 + x_4 + 1 + x_3 \\ y_4 &= 1 = x_3 + x_3 + x_3x_4 + 1 + x_3 + 1 \end{aligned}$$

Durch einsetzten von $x_3 = 1, x_4 = 0$ sind alle Gleichungen erfüllt. Unsere entschlüsselte Nachricht des Chiphertextes $y = (1, 0, 0, 1)$ ist also $x = (1, 1, 1, 0)$.

4.3 Matsumoto-Imai Scheme A (MIA)

Das *Matsumoto-Imai Scheme A* wurde erstmals 1985 unter dem Namen C^* vorgestellt[19]. Zu Ehren der Erfinder wurde das Schema später auch nach ihnen benannt. Das Schema ist auf einem Teilkörper $\mathbb{F}$ und einem Erweiterungskörper $\mathbb{E}$ definiert. Wie in Sektion 2.4 schon beschrieben, benutzen wir die Bijektion $\phi\colon \mathbb{E} \to \mathbb{F}^n$ um Elemente zwischen dem Erweiterungskörper $\mathbb{E}$ und dem Vektorraum $\mathbb{F}^n$ umzuwandeln.

C^* benutze erstmalig *branching* [20], da es aber Angriffe auf branching-Methoden gibt, macht es das Verfahren nicht sicherer. Deswegen führen wir das Verfahren ohne diese Modifikation ein.

Sei $\mathbb{F}$ ein endlicher Körper mit $p^m = q$ Elementen und $\mathbb{E}$ der Erweiterungskörper vom Grad n von $\mathbb{F}$. Die öffentlichen Polynome über $\mathbb{F}$ haben die reguläre Form

$$p_i(x_1, ..., x_n) \coloneqq \sum_{i \leq j \leq k \leq n} \gamma_{i,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{i,j} x_j + \alpha_i \text{ für } 1 \leq i \leq n$$

Die privaten Polynome sind jedoch über $\mathbb{E}$ definiert und haben folgende Form:

$$P'(X') \coloneqq (X')^{q^\lambda + 1} \text{ mit } X' \in \mathbb{E}$$

Obwohl MIA über $\mathbb{E}$ hohe Exponenten verwendet, sind die Polynome über $\mathbb{F}$ vom Grad 2. Das liegt daran, dass für jede Zahl $\lambda \in \mathbb{N}, x \mapsto x^{q^\lambda}$ eine lineare Funktion in $\mathbb{E}$ ist. Die Ganzzahl λ wird im privaten Polynom so gewählt, das $\gcd(q^\lambda + 1, q^n - 1) = 1$ für $0 < \lambda < n$ gilt. Diese Bedingung versichert dass die Polynome P' eine Inverse haben, die gegeben ist durch

$$P'(X') \coloneqq (X')^h$$

wobei $h \in \mathbb{N}$ die Gleichung $h(1 + q^\lambda) = 1 \mod (q^n - 1)$ erfüllt. [**?**]

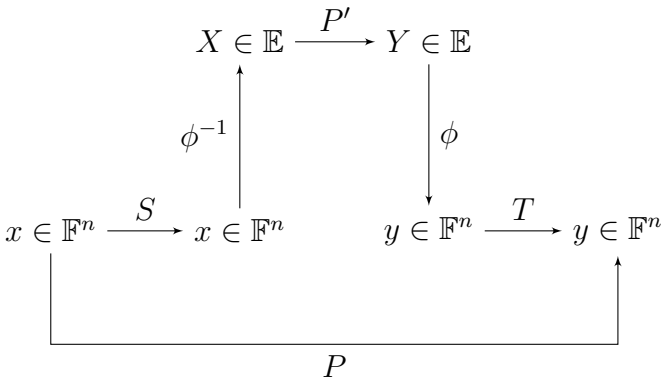

Abbildung 4: Aufbau des privaten Schlüssels beim Verfahren MIA/HFE

MIA wurde 1995 von Patarin gebrochen, durch eine sogenannte Linearization Attacke. Dabei werden grob gesagt Paare (x, y) berechnet, für die $P(x) = y$ gilt. Diese Paare bilden ein lineares Gleichungssystem für die geheimen Koeffizienten, in welches ein Chiphertext y^* eingesetzt werden kann, um den dazugehörigen Klartext x^* zu berechnen[21].

Doch die Idee einen Erweiterungskörper zu benutzen, wurde durch die *Hidden Field Equations* weitergeführt.

4.4 Hidden Field Equations (HFE)

Das *Hidden Field Equations* Schema wurde 1996 von Patarin als Verbesserung des MIA Verfahren vorgestellt[22]. Wir haben weiterhin unseren Teilkörper $\mathbb{F}$ mit $q := |\mathbb{F}|$ Elementen, den Erweiterungskörper $\mathbb{E}$ vom Grad n und die Bijektion $\phi : \mathbb{E} \to \mathbb{F}^n$ zwischen dem Erweiterungskörper und dem dazugehörigen Vektorraum. Statt nur ein einzelnes Monom verwendet HFE ein Polynom über $\mathbb{E}$. Die Ganzzahl d bestimmt den Grad des Polynoms.

$$P'(X') := \sum_{\substack{0 \leq i,j \leq d \\ q^i + q^j \leq d}} C'_{i,j} X'^{q^i + q^j} + \sum_{\substack{0 \leq k \leq d \\ q^k \leq d}} B'_k X'^{q^k} + A'$$

Da die affinen Abbildungen S und T über $\mathbb{F}^n$ definiert sind, brauchen wir unsere Abbildung ϕ wodurch der öffentliche Schlüssel folgendermaßen aufbebaut ist:

$$P = S \circ \phi^{-1} \circ P' \circ \phi \circ T$$

Um eine Nachricht zu entschlüsseln, wird die Inverse der privaten Funktion P' benötigt. Ein Algorithmus der $P'^{-1}(Y) = X$ über den Erweiterungskörper $\mathbb{E}$berechnen kann, ist der Berlekamp Algorithmus. Dabei wird das Polynom faktorisiert. Die Laufzeit des Algorithmus ist stark abhängig vom Grad d des HFE-Polynoms. Deswegen darf d nicht zu groß sein, damit das Verschlüsseln effizient bleibt. Allerdings haben Kipnis und Shamir durch die MinRank Methode einen Weg gefunden, den privaten Schlüssel zu erhalten, wenn d klein genug ist. Eine praktische und sichere Implementierung von HFE ist daher nicht möglich[13].

Durch die Erweiterung vom Monom zum Polynom in der privaten Funktion, geht die Bijektivität von MIA verloren[6]. Wir müssen dem System Redundanz durch einen Hash oder ähnlichem hinzufügen, damit eine verschlüsselte Nachricht eindeutig wiederhergestellt werden kann.

Aus kryptoanalytischer Sicht ist HFE nicht mehr sicher[23]. Durch welche Änderungen die Sicherheit des Verfahrens erhöht werden kann, sehen wir in den folgenden Sektionen.

Beispiel 20. Wir wollen die Nachricht $x = (1, 0)$ durch HFE verschlüsseln. Unser Erwei-

terungskörper ist hier $\mathbb{E} \coloneqq \mathbb{F}_{2^2}$ wie im Beispiel 7. Die Matrix-Darstellungen der affinen Abbildungen S, T sind $S \coloneqq (M_s, v_s), T \coloneqq (M_t, v_t)$ für

$$M_s \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, v_s \coloneqq \begin{pmatrix} 0 \\ 1 \end{pmatrix}, M_t \coloneqq \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, v_t \coloneqq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Das private Polynom P' ist gegeben durch $P'(X) \coloneqq X^2 + 1$. Zuerst wenden wir S an:

$$M_s x + v_s = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Das Ergebnis transferieren wir von $\mathbb{F}_2^2$ zu $\mathbb{E}$ durch $\phi^{-1}(1,1) = X + 1$. Nun wenden wir unser privates Polynom P' an:

$$P(X+1) = (X+1)^2 + 1 = X + 1$$

Wir wandeln das Ergebnis zurück zu $\mathbb{F}_2^2$ durch $\phi(X+1) = (1,1)$ Als letzes nutzen wir unsere affine Abbildung T um den Chiffretext y zu erhalten:

$$M_t(1,1) + v_t = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Unsere verschlüsselte Nachricht ist damit $y = (0,1)$.

4.5 Zusammenfassung der Falltüren

Alle Falltüren die wir bisher besprochen haben sind sehr alt. MIA wurde bereits 1985 eingeführt, STS 1993, HFE 1996 und UOV 1997 als OV und 1999 als UOV. Diese vier Schemen sind nicht die einzigen möglichen Falltüren, jedoch basieren die meisten $\mathcal{MQ}$-Public-Key Systemen auf diesen vier. In Sektion 6.3 werden wir ein neues Verfahren besprechen, welches auf Matrizenmultiplikation basiert.

Schemen die nur auf einem Körper definiert sind wie STS und UOV werden in der Literatur auch unter *small field* zusammengefasst. Schemen wie MIA und HFE die auch einen Erweiterungskörper benutzten werden auch als *big field* bzw. *medium field* bezeichnet. In Abbildung 5 ist die Taxonomie der grundlegenden Falltüren abgebildet.

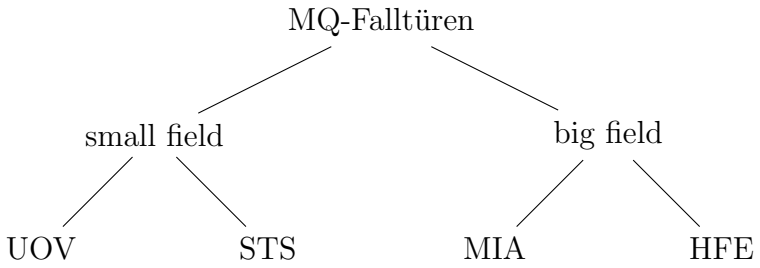

Abbildung 5: Taxonomie der $\mathcal{MQ}$-Falltüren

Außer UOV mit geeigneten Parametern wurden alle anderen Falltüren in ihrer ursprünglichen Form erfolgreich angegriffen und gebrochen. Im nächstem Kapitel besprechen wir, durch welche Modifikationen die Sicherheit der Verfahren erhöht werden kann.

5 Modifikationen

Die meisten der generellen Falltüren sind nicht mehr sicher und es gibt verschiedene Angriffe, um diese Verfahren zu brechen. Durch verschiedene Modifikationen an diesen Schemen lässt sich die Sicherheit jedoch erhöhen. Diese Modifikationen sind generell und theoretisch auf alle Falltüren anwendbar. Für manche Schemen gibt es jedoch effizientere Modifikationen.

5.1 Plus Methode „+“

Wie der Name schon verrät, werden bei dieser Methode zusätzliche Gleichungen hizugefügt. Diese zusätzlichen zufällig gewählten Gleichungen können als Störfaktor gesehen werden, die Angreifer am entschlüsseln hindern sollen.

Seien $\tilde{P} \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^{\tilde{m}})$ die ursprünglichen Gleichungen. Dann werden $a \in \mathbb{N}$ zufällige quadratische Gleichungen $(\pi_1, ..., \pi_a) \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^a)$ ohne eine Falltür zu den ursprünglichen Gleichungen hinzugefügt, wodurch das private Gleichungssystem $P' \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$ entsteht. Hierbei ist $m \coloneqq \tilde{m} + a$ für $m, \tilde{m} \in \mathbb{N}$.

Damit bilden die Polynome $p_1, ..., p_m$ den neuen geheimen Schlüssel P'.

$$
\begin{aligned}
p'_1(x'_1, ..., x'_n) &\coloneqq \tilde{p}_1(x'_1, ..., x'_n) \\
&\vdots \\
p'_{\tilde{m}}(x'_1, ..., x'_n) &\coloneqq \tilde{p}_{\tilde{m}}(x'_1, ..., x'_n) \\
p'_{\tilde{m}+1}(x'_1, ..., x'_n) &\coloneqq \pi_1(x'_1, ..., x'_n) \\
&\vdots \\
p'_m(x'_1, ..., x'_n) &\coloneqq \pi_a(x'_1, ..., x'_n)
\end{aligned}
$$

Die Plus-Methode war als Verbesserung der Schemen MIA und HFE gedacht, jedoch hat sich gezeigt, dass die Sicherheit der Verfahren dadurch nicht erhöht. Hinzu kommt dass der Rechenaufwand um q^a erhöht wird, da nur q^{-a} von allen Lösungen des originalen Schlüssels $\tilde{P}$ auch eine Lösung der a Gleichungen $(\pi_1, ..., \pi_a)$ sind. [6]

5.2 Minus Methode „-“

Bei der Minus-Methode werden Gleichungen entfernt. Diese Methode wurde erstmals 1994 von Amir Shamir vorgestellt[25]. Obwohl sie relativ einfach erscheint, kann sie sehr wirksam gegen einige Angriffe sein.

Für diese Modifikation wählen wir $\tilde{m} \coloneqq m - r$ für ein $r \in \mathbb{N}$ und fügen unseren Public-Key Gleichungen eine Reduzierungsfunktion $R\colon \mathbb{F}^m \to \mathbb{F}^{\tilde{m}}$ hinzu. Unser öffentlicher Schlüssel P sieht folgendermaßen aus: $P \coloneqq R \circ T \circ P' \circ S$. Die Reduzierungsfunktion R ist definiert als $R(y_1, ..., y_n) \coloneqq (y_1, ..., y_{m-r})$. Dabei werden die letzten r Komponenten der verschlüsselten Nachricht y weggelassen.

$$
\begin{aligned}
p_1(x_1, ..., x_n) &\to p_1(x_1, ..., x_n) \\
&\vdots \\
p_{m-r}(x_1, ..., x_n) &\to p_{m-r}(x_1, ..., x_n) \\
p_{m-r+1} \quad (x_1, ..., x_n) & \\
\vdots & \Big\} \text{ verworfen} \\
p_m \quad (x_1, ..., x_n) &
\end{aligned}
$$

Besonders nützlich ist die Minus-Methode für Digitale Signaturen. Es gibt keine Performance Einschränkungen, da ein Dokument keine eindeutige Signatur benötigt[13]. Da die Verschlüsselungen eindeutig sein müssen, müsste beim Invertieren von $P(x)$ die fehlenden Werte geraten werden. Dies hat einen Zeitverlust beim Entschlüsseln zur Folge.

C^* wird durch die Minus-Modifikation erheblich schwerer zu lösen. Die auf C* basierende Public-Key Verfahren SFLASH ist eine C^{*-} Instanz, die von der „New European Schemes for Signatures, Integrity and Encryption“ (NESSIE) akzeptiert wurde. Jedoch fanden Shamir und andere 2007 eine Möglichkeit diese zu brechen. Auch bei HFE findet diese Methode eine Anwendung, was wir in Sektion 6.2 sehen werden.

5.3 Branching Methode „⊥“

Wie bereits in Sektion 4.3 erwähnt, wurde MIA erstmals mit einer Branching Methode eingeführt. Generell gesagt wird beim Branching das private Polynom P' partitioniert. Die Zweige sind dabei komplett unabhängig voneinander

Im folgenden Bild sehen wir eine grapfische Darstellung einer MQ-Falltür mit zwei Zweigen. Hierbei sind $n = n_1 + n_2$ und $m = m_1 + m_2$ für $n_1, n_2, m_1, m_2 \in \mathbb{N}$.

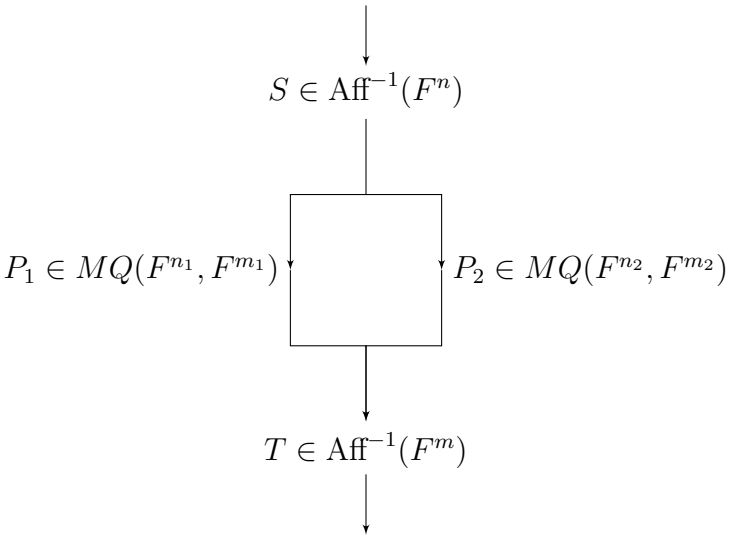

Abbildung 6: Darstellung einer MQ-Trapdoor mit Branching

Das Ziel beim Branching ist im Gegensatz zu anderen Modifikationen nicht die Erhöhung der Sicherheit, sondern der Performance. Da die einzelnen Zweige unabhängig voneinander sind, können diese parallel berechnet werden, was zu einem Speedup führt[6]. Jedoch fand Patarin 1995 einen Angriff, dessen Algorithmus die einzelnen Zweige trennt, um das MIA-Schema noch effizienter anzugreifen [21].

Die Branching Modifikation ist daher nicht sicher und sollte nicht verwendet werden, obwohl sie die Performance des Systems erhöht.

5.4 Vinegar Variablen „v“

Die Idee, Vinegar Variablen außerhalb des UOV Schemas zu benutzen, wurde erstmals im Zusammenhang mit HFE benutzt[17]. Hierbei werden die linearen und konstanten Terme des privaten Polynoms mit zufälligen Vinegar-Variablen vermischt. Die dadurch

verursachte Störung erhöht die Sicherheit gegen bestimmte Angriffe.

Das private Polynomsystem $P' \colon \mathbb{E} \to \mathbb{E}$ wird ersetzt durch $P' \colon \mathbb{E} \times k^v \to \mathbb{E}$. Hier werden v Vinegar-Variablen dem HFE System hinzugefügt. Die Form des HFEv-Polynoms ist

$$P'(X, x_1, ..., x_v) = \sum_{\substack{0 \leq i,j \leq d \\ q^i + q^j \leq d}} C'_{i,j} X'^{q^i + q^j} + \sum_{\substack{0 \leq k \leq d \\ q^k \leq d}} B'_k X'^{q^i} + \sum_{\substack{0 \leq k \leq d \\ q^k \leq d}} A'_k \Omega_k(x_1, ..., x_v) X'^{q^i} + \Phi(x_1, ..., x_v)$$

wobei $A_i, B_i, C_{i,j} \in \mathbb{E}$ sind. Die Funktionen $\Omega_k \colon \mathbb{F}^v \to \mathbb{E}$ und $\Phi \colon \mathbb{F}^v \to \mathbb{E}$ sind linear bzw. quadratisch über den Vinegar-Variablen. Dadurch bleibt das gesamte System höchstens quadratisch. Die Variablen die in X stecken sind unsere Oil-Variablen und $(x_1, ..., x_v)$ die Vinegar-Variablen. Die Vinegar-Variablen werden beim Signieren zufällig zugeteilt. Das dabei entstehende Polynom wird wie ein reguläres HFE-Polynom invertiert.

Beim Entschlüsseln muss die ursprüngliche Falltür q^v mal invertiert werden, um die private Gleichung zu invertieren. Somit findet man die richtige Lösung, die den Vinegar Variablen entspricht. Bei Verschlüsselungsverfahren darf q^v deshalb nicht zu groß sein, da der Rechenaufwand sonst zu hoch ist.

Da jede Lösung eine gültige Signatur ist, gibt es keinen höheren Rechenaufwand für Signaturverfahren[6].

5.5 Innere Pertubationen „i“

Perturbation kann als Störung angesehen werden, die wir dem System hinzufügen. So kann man die Vinegar-Methode auch als äußere Perturbation bezeichnen. Bei der inneren Perturbation wird keine zusätzliche Information benötigt, sondern durch die vorhanden Variablen erschaffen. Es wird ein Unterraum durch die Variablen aufgespannt, der dem privaten Schlüssel Störung hinzufügt. Die Methoden hat sich als besonders nützlich im Zusammenhang mit MIA und HFE erwiesen[13].

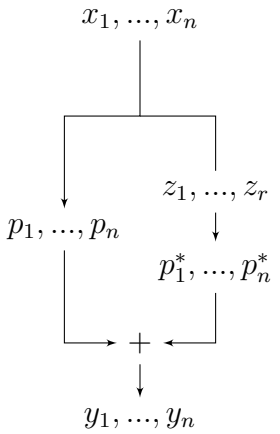

Abbildung 7: $\mathcal{MQ}$-Falltür mit innerer Perturbation

Dafür wählen wir $w \in \mathbb{N}$ zufällige lineare Funktionen

$$z_1(x_1, ..., x_n) = \sum_{j=1}^{n} a_{j,1} x_j + \beta_1$$
$$\vdots$$
$$z_w(x_1, ..., x_n) = \sum_{j=1}^{n} a_{j,w} x_j + \beta_w$$

Die Lösung dieser Gleichungen sind wiederum die Variablen von m zufälligen Polynomen vom Grad 2.

$$P^*(z_1, ..., z_w) = (p_1^*(z_1, ..., z_w), ..., p_n^*(z_1, ..., z_w))$$

Die Perturbationspolynome werden einfach mit dem üblichen privaten Polynomen addiert, wodurch der öffentliche Schlüssel folgende Form hat: $P = S \circ (P' + P^*) \circ T$. Die privaten Polynome sehen wie folgt aus:

$$P' + P^* := \begin{cases} p'_1 := p_1(x'_1, ..., x_n) + p_1^*(z'_1, ..., z'_w) \\ \vdots \\ p'_m := p_m(x'_1, ..., x_n) + p_m^*(z'_1, ..., z'_w) \end{cases}$$

Die Rechenzeit beim Invertieren der zentralen Polynome erhöht sich um einen Faktor

von q^w. Für jede Lösung von P' muss überprüft werden, ob die w Polynome von P^* mit der Lösung übereinstimmt.

Innere Perturbation erhöht die Sicherheit von MIA und HFE gegen bestimmte Angriffe. Jedoch gibt es immer noch einige erfolgreiche Angriffe auf die perturbierten Schemen. Deswegen wurde diese auch noch mit der Plus-Methode verknüpft. PMI+ (Perturbed Matsumi Imai) und IPHFE+ (Internal Perturbed Hidden Field Equations) gelten derzeit als sicher. Der erhöhte Entschlüsselungsaufwand der durch die beiden Modifikationen entsteht, macht die beiden Verfahren jedoch zu ineffizient für einen praktischen Gebrauch[14].

5.6 Zusammenfassung der Modifikationen

In diesem Kapitel haben wir Modifikationen von $\mathcal{MQ}$-Falltüren behandelt, welche die Sicherheit oder Effizienz dieser Schemen erhöhen sollen. Es gibt außerdem auch weiter Methoden Falltüren zu modifizieren, die wir nicht im Detail behandeln werden.

Analog zur Minus-Methode gibt es auch die Fixing-Methode. Dabei wird nicht die Anzahl der Gleichungen reduziert sondern die der Variablen. Diese Variante verlangsamt das Verfahren, da die weggelassenen Variablen trotzdem zum Klartext passen müssen.

Umgekehrt zum Fixing, können wir nicht nur Variablen entfernen, sondern auch hinzufügen. Dies passiert durch die Mask-Methode. Hier werden Variablen hinzugefügt.

Eine weitere Modifikation sind „sparse polynomials“ (zu deutsch: spärliche/karge Polynome). Dabei werden Polynome mit wenigen Termen verwendet, wodurch Rechenzeit gespart wird. Alle drei Methoden wurde aus kryptoanalytischer Sicht nicht genug getestet, um entgültige Aussagen bezüglich ihrer Sicherheit zu treffen[6].

Modifikationen	Symbol	Sicherheit
Plus	$+$	sicher
Minus	$-$	sicher
Branching	$\perp$	unsicher
Vinegar	v	sicher
Innerer Perturbation	i	sicher
Fixing	f	offen
Mask	m	offen
Sparse Polynomials	s	offen

Tabelle 1: Modifikationen und ihre Sicherheit

Im folgenden Kapitel betrachten wir, welchen Einfluss die besprochenen Modifikationen auf die MQ-Schemen haben und stellen vielversprechende Schemen vor, die bis zum jetzigem Zeitpunkt als sicher gelten.

6 Vielversprechende Schemen

Bis auf UOV mit geeigneten Parametern sind alle grundlegenden Falltüren nicht mehr sicher. Im Folgenden stelle ich die Verfahren vor, die bis zum jetzigem Zeitpunkt als sicher gelten. Unter anderem geschieht dies, durch die im vorherigem Kapitel besprochenen Modifikationen.

6.1 Rainbow

Das „Rainbow"-Schema wurde von Jintai Ding und Dieter Schmidt 2005 vorgestellt[26] und kann als eine Version von UOV angesehen werden, die mehrere Schichten benutzt. Sowohl die Schlüssel- und Signaturgröße konnte reduziert, als auch die Performance verbessert werden[14].

Sei V die Menge $\{1, 2, 3, ..., n\}$ und $v_1, ..., v_u$ eine Menge von u Ganzzahlen sodass $0 < v_1 < ... < v_u = n$ gilt. Wir definieren die Mengen $V_l = \{1, 2, ..., v_l\}$ für $l = 1, ..., u$. Für diese Mengen gilt

$$V_1 \subset V_2 \subset \cdots \subset V_u = V$$

und dass v_i die Anzahl der Elemente von S_i ist. Für $i = 1, ..., u-1$ sei $o_i = v_{i+1} - v_i$ und $O_i = S_{i+1} - S_i$, sodass o_i die Anzahl der Elemente von O_i ist. Seien P_l nun alle Polynome der Form

$$y_i := \sum_{j \in V_l} \sum_{k \in V_l} \delta_{i,j,k} x_j x'_k + \sum_{j \in V_l} \sum_{k \in O_l} \gamma_{i,j,k} x'_j x'_k + \sum_{i \in V_l \cup O_l} \beta_{i,j} x_j + \alpha'_i$$

In diesen Polynomen sind alle x_j Vinegar Variablen wenn $j \in V_l$ und x_k sind Oil-Variablen wenn $k \in O_l$. Außerdem gehören die Variablen zur l-ten Schicht. P_l beinhaltet alle Polynome der l-ten Schicht. Die Vinegar Variablen der $(l+1)$-ten Schicht sind alle Variablen der l-ten Schicht da

$$V_{i+1} = O_i \cup V_i$$

Seien $P_l = (F_{l,1}, ..., F_{l,o_i})$ für $l = 1, ..., u-1$ alle Polynome der l-ten Schicht wobei $F_{l,i}$ ein zufällig gewähltes Polynom von P_l ist. Unser geheimes Polynomsystem ist nun $P' \colon \mathbb{F}^n \to \mathbb{F}^{n-v_1}$ mit $P' = (P_1, ..., P_{u-1})$. P' hat also $u-1$ Schichten. Die erste Schicht

besteht aus o_1 Polynomen $P_{1,1}, ..., P_{1,o_1}$ mit den Oil-Variablen $\{x_i | i \in O_1\}$ und den Vinegar-Variablen $\{x_j | j \in V_j\}$.

Durch die Schichten baut sich ein „Regenbogen“ aus Variablen auf:

$$\begin{aligned}
&\{x_1, ..., x_{v_1}\}, \{x_{v_1+1}, ..., x_{v_2}\};\\
&\{x_1, ..., x_{v_1}, x_{v_1+1}, ..., x_{v_2}\}, \{x_{v_2+1}, ..., x_{v_3}\};\\
&\qquad\qquad\vdots\\
&\{x_1, ..., ..., ..., ..., ..., ..., ..., x_{v_{u-1}}\}, \{x_{v_{u-1}+1}, ..., x_n\};
\end{aligned}$$

Um eine Nachricht zu signieren müssen wir, wie übliche, das private Polynomsystem invertieren. Dafür wenden wir den gleichen Algorithmus, wie bei UOV, für jede Schicht an. Wir weisen den Vinegar-Variablen der ersten Schicht zufällige Werte zu und berechnen die o_1 Oil-Variablen in den o_1 Polynomen durch den Gauß-Algorithmus. Da die Vinegar-Variablen der nächsten Schicht alle Variablen der vorherigen Schicht sind, setzen wir diese nun ein und berechnen wieder die Oil-Variablen.

Dies wiederholen wir, bis alle Unbekannten berechnet wurden. Hat das Gleichungssystem einer Schicht keine Lösung, beginnen wir bei der Zuweisung der ersten Schicht[13].

Rainbow kann als Verallgemeinerung von UOV angesehen werden. So ist UOV eine Rainbow Instanz, die aus nur einer Schicht besteht. Beide Verfahren gelten bis heute als sicher und vor allem das Rainbow Schema ist ein aussichtsreiches Verfahren für digitale Signaturen[14].

6.2 HFEv-

HFEv- ist eine Variante der Hidden Field Equations bei der Vinegar Variablen benutzt werden, sowie die Minus Methode. Da wir alle Komponenten schon vorgestellt haben, gehen wir nicht nochmal im Detail auf die Methoden ein. Das private Polynomensystem hat die Form

$$P'(X, x_1, ..., x_v) = \sum_{\substack{0 \leq i,j \leq d \\ q^i + q^j \leq d}} C'_{i,j} X'^{q^i + q^j} + \sum_{\substack{0 \leq k \leq d \\ q^k \leq d}} B'_k X'^{q^i} + \sum_{\substack{0 \leq k \leq d \\ q^k \leq d}} A'_k \Omega_k(x_1, ..., x_v) X'^{q^i} + \Phi(x_1, ..., x_v)$$

Eine Implementierung für digitale Signaturen von HFEv- ist QUARTZ, welches 2001 von Patarin und Courtois vorgestellt wurde. Die Parameter die dabei verwendet wurden sind

$$(\mathbb{F}, n, d, a, v) = (\mathbb{F}_2, 103, 129, 3, 4)$$

Damit wird in diesem Schema vom Körper $\mathbb{F}^{107}$ nach $\mathbb{F}^{100}$ abgebildet. Der öffentlichen Schlüssels ist 71kB und der private 3kB groß. Das QUARTZ Schema ist dennoch langsam. Es dauert ungefähr 9 Sekunden eine Signatur zu signieren[13].

6.3 SimpleMatrix

Die SimpleMatrix Methode (auch ABC Methode genannt) wurde 2013 erstmals vorgestellt und ist damit ein relativ neues Verfahren. Es eignet sich für Verschlüsselungen[27]. SimpleMatrix widersteht außerdem der minRank Attacke, welche wirksam gegen MIA und HFE ist.

Seien $n, m, s \in \mathbb{N}$ Ganzzahlen sodass $n = s^2$ und $m = 2n$ gilt.

$$A = \begin{pmatrix} x_1 & x_2 & \cdots & x_s \\ x_{s+1} & x_{s+2} & \cdots & x_{2s} \\ \vdots & \vdots & & \vdots \\ x_{n-s+1} & x_{n-s+2} & \cdots & x_n \end{pmatrix}$$

$$B = \begin{pmatrix} b_1 & b_2 & \cdots & b_s \\ b_{s+1} & b_{s+2} & \cdots & b_{2s} \\ \vdots & \vdots & & \vdots \\ b_{n-s+1} & b_{n-s+2} & \cdots & b_n \end{pmatrix}$$

$$C = \begin{pmatrix} c_1 & c_2 & \cdots & c_s \\ c_{s+1} & c_{s+2} & \cdots & c_{2s} \\ \vdots & \vdots & & \vdots \\ c_{n-s+1} & c_{n-s+2} & \cdots & c_n \end{pmatrix}$$

A, B und C sind drei $s \times s$ Matrizen, wobei $x_i \in \mathbb{F}$. b_i und c_i sind zufällig gewählte Kombinationen der Elemente der Menge $\{x_1, ..., x_n\}$ für $i = 1, ..., n$. Wir definieren die $s \times s$ Matrizen E_1 und E_2 als $E_1 = AB, E_2 = AC$. Die m privaten Polynome in P' bestehen aus den m Komponenten von E_1 und E_2.

Der private Schlüssel besteht damit aus den affinen Abbildungen S, T und den beiden Matrizen B und C. Der öffentliche Schlüssel ist wie gewohnt $P = S \circ P' \circ T \colon \mathbb{F}^n \to \mathbb{F}^m$.

Um einen Klartext x zu verschlüsseln, wenden wir den öffentlichen Schlüssel wie bei anderen Schemen an: $P(x) = y$. Der Chiffretext y wird entschlüsselt, indem wir P' invertieren. Dabei wird ein lineares Gleichungssystem gelöst, welches wir durch die Matrizen A, B und dem Chiffretext erlangen.

Die Verschlüsselung kann fehlschlagen, wenn der Rank der Matrix A geringer ist als $s - 2$. Allerdings tritt dieser Fall schon bei $\mathbb{F}_q, q = 2^8, s = 8$ mit einer Wahrscheinlichkeit von 10^{-22} auf[27].

7 Zusammenfassung und Ausblick

Das Problem, welches den $\mathcal{MQ}$-Systemen Sicherheit verleiht, ist das Lösen von zufällig gewählten, multivariaten Gleichungen über endlichen Körpern. Dieses NP-vollständige Problem soll auch nach Einführung von Quantencomputern schwer zu lösen bleiben[28].

Die Gruppe der *small field* Schemen versteckt die Falltür über einen einfachen endlichen Körper. Dazu gehören UOV, die mehrschichtige Rainbow Variante sowie die STS Schemen. Auch das neu entwickelte SimpleMatrix Verfahren kann dazu gezählt werden.

Der Begriff *big field* umfasst alle Falltüren, die den privaten Schlüssel über einen Erweiterungskörper definieren. Die grundlegenden dazugehörigen Verfahren sind MIA und HFE. Hinzu kommen alle Varianten wie HFEv-, C^{*-} oder PMI+ und IPHFE+.

Eine Vielzahl von Modifikationen hilft die Sicherheit der Systeme gegenüber bestimmter Angriffe zu erhöhen. Meistens geschieht dies auf Kosten der Performance. Die Sicherheit der Verfahren kann in den seltensten Fällen mathematisch bewiesen werden. Die Erfahrung durch kryptoanalytische Untersuchungen unterstützen jedoch die Sicherheitsannahmen.

Ein klarer Vorteil der $\mathcal{MQ}$-Signaturverfahren sind die kurzen Signaturen von ein paar hundert Bits. Kein anderes Postquantum Schema kann dies unterbieten. Vor allem Rainbow ist ein effizientes Verfahren für digitale Signaturen, welches kryptographischen Angriffen für lange Zeit standgehalten hat. Jedoch haben auch alle Verfahren den Nachteil, dass die öffentlichen Schlüssel vergleichsweise groß sind. Die Größe variiert von 10kB bis 100kB [14]. Ein RSA Schlüssel hat im Vergleich eine Größe von 1024Bit[5].

Obwohl seid 30 Jahren an Public-Key Verfahren mit multivariaten quadratischen Gleichungen geforscht wird, werden immer noch neue Verfahren und Angriffe entdeckt. Doch gerade neue Methoden müssen weiterhin erforscht werden, um eine endgültige Aussage über ihre Sicherheit und Verwendungsmöglichkeiten in der Post-Quanten-Kryptographie zu treffen.

Trotzdem sind unter den Kandidaten für Public-Key Kryptographie, die auch nach der Einführung von Quantencomputern sicher bleiben sollen, die multivariaten quadratischen Gleichungen eine vielversprechende Lösung.

Literatur

[1] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[2] National Institute of Standards and Technology (NIST). FIPS PUB 186: Digital Signature Standard (DSS), 1994.

[3] National Institute of Standards and Technology (NIST). FIPS PUB 186-4: Digital Signature Standard (DSS), 2013.

[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE-IT*, 22:644–654, 1976.

[5] Johannes Buchmann. *Einführung in die Kryptographie*. Springer-Lehrbuch, Berlin, 2016. 6. Auflage.

[6] Christopher Wolf. Multivariate Quadratic Polynomials in Public Key Cryptography, 2005.

[7] Christian Karpfinger und Kurt Meyberg. *Algebra*. Springer-Verlag Berlin Heidelberg, 2010. 2. Auflage.

[8] James McKernan. Abstract Algebra. University Lecture, 2015. `http://www.math.ucsd.edu/~jmckerna/Teaching/15-16/Spring/103B/l_17.pdf`.

[9] Siegfried Bosch. *Algebra*. Springer-Verlag Berlin Heidelberg, 2009. 1. Auflage.

[10] Albrecht Beutelspacher. *Lineare Algebra*. Springer Spektrum, 2014. 8. Auflage.

[11] Marcel Berger. *Geometry I*. Springer-Verlag Berlin Heidelberg, 1987. 1. Auflage.

[12] Jintai Ding and Bo-Yin Yang. Multivariate public key cryptographie, 2009. Pages 193-242 in Post-quantum cryptography.

[13] Jintai Ding, Jason Gower, and Dieter Schmidt. *Multivariate Public Key Cryptosystems*, volume 25 of *Advances in Information Security*. Springer, 2006.

[14] Jintai Ding and Albrecht Petzoldt. Current State of Multivariate Cryptographie. *IEEE Security & Privacy*, 15:28–36, 2017.

[15] Jacques Patarin. The oil and vinegar signatur scheme. Im Dagstuhl Workshop für Kryptographie vorgstellt (Overheadfolien), 1997.

[16] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[17] A. Kipnis and J. Patarin and L. Goubin. Unbalanced Oil and Vinegar Signatur Schemes, 1999.

[18] Christopher Wolf, An Braeken, and Bart Preneel. Efficient Cryptanalysis of RSE(2)PKC and RSSE(2)PKC. Cryptology ePrint Archive, Report 2004/237, 2004. `https://eprint.iacr.org/2004/237`.

[19] Hideki Imai and Tsutomu Matsumoto. Algebraic methods for constructing asymmetric cryptosystems. In Jacques Calmet, editor, *Algebraic Algorithms and Error-Correcting Codes*, pages 108–119, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[20] T. Matsumoto and H. Imai. PUBLIC QUADRATIC POLYNOMIAL-TUPLES FOR EFFICIENT SIGNATURE-VERIFICATION AND MESSAGE-ENCRYPTION, 1988.

[21] J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Kex Scheme of Eurocrypt'88. In *Advances in Cryptology — CRYPTO' 95*, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[22] Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 33–48, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[23] Nicolas T. Courtois. On multivariate signature-only public key cryptosystems. Cryptology ePrint Archive, Report 2001/029, 2001. `https://eprint.iacr.org/2001/029`.

[24] Christopher Wolf and Bart Preneel. Asymmetric cryptography: Hidden field equations. Cryptology ePrint Archive, Report 2004/072, 2004. `https://eprint.iacr.org/2004/072`.

[25] Adi Shamir. Efficient Signature Schemes Based on Birational Permutations. In *Advances in Cryptology — CRYPTO' 93*, pages 1–12, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[26] Jintai Ding and Dieter Schmidt. Rainbow, a New Multivariable Polynomial Signature Scheme. In *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[27] Chengdong Tao, Adama Diene, Shaohua Tang, and Jintai Ding. Simple Matrix Scheme for Encryption. In *Post-Quantum Cryptography*, pages 231–242, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[28] Daniel Bernstein and Johannes Buchmann und Erik Dahmen. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009.

Abbildungsverzeichnis

Tabellenverzeichnis