Bachelorarbeit

Post-Quanten-Kryptographie: Codebasierte Verfahren

Tobias Nießen

Matrikelnummer 3222880

Institut für Theoretische Informatik

Fakultät für Elektrotechnik und Informatik

Leibniz Universität Hannover

3. September 2018

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht habe, und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen hat.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Verwendete Notationen

Die folgenden Abkürzungen und Notationen werden im Rahmen dieser Arbeit verwendet. Alle Vektoren werden als Spaltenvektoren aufgefasst.

Mengenoperationen

$\mathcal{P}(X)$ Potenzmenge von X

$\mathcal{P}_k(X)$ Menge der k-elementigen Teilmengen von X

Vektor- und Matrixoperationen

$\mathbf{a} \,\|\, \mathbf{b}$ Konkatenation der Vektoren $\mathbf{a}$, $\mathbf{b}$

$\langle \mathbf{a}, \mathbf{b} \rangle$ Skalarprodukt der Vektoren $\mathbf{a}$, $\mathbf{b}$

$\mathbf{a} \oplus \mathbf{b}$ Bitweise Exklusiv-Oder-Operation zwischen Vektoren $\mathbf{a}, \mathbf{b}$ über $\mathbb{F}_2$

$M^\intercal$ Transponierte einer Matrix

$M \cdot \mathbf{v}$ Matrix-Vektor-Produkt der Matrix M und des Vektors $\mathbf{v}$

$\mathbf{v} \cdot M$ Abkürzung für $M^\intercal \cdot \mathbf{v}$

$\ker(M)$ Kern der durch M definierten linearen Abbildung

1 Einleitung

Die mögliche Konstruktion von Quantencomputern erhielt spätestens dann breite Aufmerksamkeit, als Peter Shor 1994 einen Algorithmus vorstellte, der — unter Zuhilfenahme eines Quantencomputers — sowohl Primfaktorzerlegung als auch die Berechnung diskreter Logarithmen in polynomieller Zeit und somit potenziell um ein Vielfaches schneller als herkömmliche Algorithmen ermöglichte [27]. Viele verbreitete kryptographische Verfahren wie beispielsweise Rivest–Shamir–Adleman (RSA), der Elliptic Curve Digital Signature Algorithm (ECDSA) und Diffie-Hellman basieren auf diesen oder ähnlichen Problemen aus dem Bereich der Zahlentheorie und bieten vor einem Angreifer mit Zugang zu einem ausreichend großen Quantencomputer keinen Schutz mehr.

Die prognostizierte Beschleunigung dieser und anderer Berechnungen hat die Entwicklung von Quantencomputern umso attraktiver für akademische und wirtschaftliche Zwecke gemacht: Weltweit forschen Wissenschaftler an Möglichkeiten, Quantencomputer zu konstruieren. In der Vergangenheit sind physikalische Umsetzungen nur sehr begrenzt gelungen, sodass Anwendungen hauptsächlich simuliert wurden, doch Prozessoren wie der von Intel im Januar 2018 angekündigte „Tangle Lake“-Prozessor, der nach Angaben des Unternehmens aus 49 Qubits besteht, übertreffen alle derzeitig möglichen Simulationen: Allein der Systemzustand zu einem Zeitpunkt würde die Speicherung von Daten im Umfang von mehreren Petabyte erfordern. Es ist ungewiss, wie schnell sich diese Entwicklung fortsetzen wird, doch sicher ist, dass die kommerzielle Erschließung dieser Technologie einen massiven Einfluss auf aktuelle und zukünftige Anwendungen haben wird.

Algorithmen wie der von Shor machen Gebrauch von sogenannter Quantenparallelität: Durch die Verwendung von Qubits anstelle klassischer Bits können Quantencomputer Berechnungen auf exponentiell vielen Zuständen gleichzeitig durchführen. So ist es beispielsweise möglich, einen Funktionswert für jeden möglichen Eingabewert gleichzeitig zu berechnen, aber es ist nicht möglich, mehr als ein Ergebnis auszulesen. Auch klassische Operationen, wie sie aus herkömmlichen elektrotechnischen Schaltungen bekannt sind, können in Quantenberechnungen abgebildet werden, da-

her sind Quantencomputer zu allen klassischen Berechnungen gleichermaßen fähig wie existierende Technologien. Erst Quantenparallelität macht den entscheidenden Unterschied aus, und es ist unbekannt, in welchem Ausmaß sich dies auf zukünftige Programme auswirken wird. Die Entwicklung von Algorithmen, welche durch Quantenparallelität Vorteile gegenüber klassischen Berechnungen haben, erweist sich als wesentlich komplexer als die Programmierung herkömmlicher Hardware, daher soll im Rahmen dieser Arbeit auf diese Thematik nicht im Detail eingegangen werden. Für eine Einführung in Quantenberechnungen sei der Leser auf „An Introduction to Quantum Computing for Non-Physicists“ von Rieffel und Polak verwiesen [26].

Während Quantencomputer insbesondere bei ausgewählten Problemen aus dem Bereich der Zahlentheorie deutliche Geschwindigkeitsvorteile zeigen, gibt es auch Probleme, die von Quantencomputern nicht schneller gelöst werden können. Neben vielen klassischen Verfahren, insbesondere symmetrischen Verschlüsselungen wie AES, sind nach bisherigem Kenntnisstand auch die aktuell weniger weit verbreiteten Klassen der hash-, gitter- und codebasierten sowie multivariaten Kryptographie selbst nach der Konstruktion ausreichend großer Quantencomputer ebenso sicher wie bisher. Gerade diesen Klassen mangelt es bislang an Bekanntheit und praktischem Einsatz, unter anderem, da sie bislang nicht mit der Effizienz klassischer Verfahren mithalten können [4].

Diese Arbeit behandelt codebasierte Kryptographie als vielversprechenden Lösungsansatz für verschiedene kryptographische Problemstellungen. Die Grundlage der codebasierten Kryptographie bildet die Codierungstheorie; fehlerkorrigierende Codes bilden wiederum die Grundlage für die in dieser Arbeit vorgestellten kryptographischen Primitiven und die damit verbundenen mathematischen Probleme, auf denen die vermutete Sicherheit der Algorithmen basiert. Aus diesem Grund werden im folgenden Kapitel zunächst notwendige Inhalte der Codierungstheorie erläutert.

In Kapitel 3 werden wichtige codebasierte Verfahren beschrieben. Dazu zählen asymmetrische Verschlüsselung nach McEliece und Niederreiter, digitale Signaturen nach Courtois, Finiasz und Sendrier, Authentisierung nach Stern und verwandte Algorithmen für kryptographische Hashfunktionen und kryptographisch sichere Pseudozufallszahlenerzeugung. Diese Verfahren gelten auch heute, teilweise mehre-

re Jahrzehnte nach ihrer Veröffentlichung, als sicher, selbst gegen Angriffe unter Zuhilfenahme von Quantencomputern. Der Fokus liegt dabei auf asymmetrischer Verschlüsselung, da die in diesem Bereich etablierten Methoden wie RSA besonders anfällig für Angriffe durch Quantencomputer zu sein scheinen.

Einen genaueren Überblick über die erhoffte Sicherheit der vorgestellten Verfahren verschafft Kapitel 4, welches bekannte Angriffe vorstellt und zeigt, weshalb codebasierte Kryptographie trotz allem eine vielversprechende Alternative zu klassischen kryptographischen Verfahren bleibt, und wie eine zu etablierten Verfahren wie RSA äquivalente Sicherheit erreicht werden kann.

2 Codierungstheorie

2.1 Fehlerkorrigierende Codes

In der Codierungstheorie dienen fehlerkorrigierende Codes zur nachträglichen Korrektur von Übertragungsfehlern. Nachrichten werden als Vektoren mit Länge k über einem endlichen Körper $\mathbb{F}_q$ für eine Primzahlpotenz q (siehe Anhang A) dargestellt und vor der Übertragung auf einen längeren Vektor mit Länge $n > k$ abgebildet. Die hinzugefügte Redundanz dient beim Empfänger zur Erkennung und Korrektur von Übertragungsfehlern, die einzelne Elemente des Vektors verändert haben [19]. Längere Nachrichten können in Blöcke der Länge k aufgeteilt und anschließend codiert und gesendet werden. Die Codierung von Nachrichten in Codewörter und die Decodierung von Codewörtern in Nachrichten ist abhängig von der Art des Codes.

Definition 2.1.1 (Fehlerkorrigierende Codes). *Sei q eine Primzahlpotenz, $\mathbb{F}_q$ ein endlicher Körper, n eine positive natürliche Zahl. Ein* fehlerkorrigierender Code *$\mathcal{C}$ ist eine Teilmenge von $\mathbb{F}_q^n$. Die Elemente von $\mathcal{C}$ werden als* Codewörter *bezeichnet.*

Übertragungsfehler werden in der Codierungstheorie durch die Addition eines Fehlervektors $\mathbf{e} \in \mathbb{F}_q^n$ zum übertragenen Codewort $\mathbf{c} \in \mathcal{C}$ dargestellt: $\mathbf{y} = \mathbf{c} + \mathbf{e}$. Im Fall eines binären Codes $\mathcal{C} \subseteq \mathbb{F}_2^n$ ist dies äquivalent zur Exklusiv-Oder-Operation $\mathbf{y} = \mathbf{c} \oplus \mathbf{e}$.

Definition 2.1.2 (Hamming-Distanz). *Seien $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}_q^n$ zwei Vektoren. Die* Hamming-Distanz $\operatorname{dist}(\mathbf{a}, \mathbf{b})$ *ist die Anzahl der Stellen, in denen sich $\mathbf{a}$ und $\mathbf{b}$ unterscheiden:*

$$\operatorname{dist}(\mathbf{a}, \mathbf{b}) \coloneqq \left| \left\{ i \in \{1, \ldots, n\} \mid a_i \neq b_i \right\} \right|$$

Wird ein Codewort $c \in \mathcal{C}$ übertragen und ein Vektor y gleicher Länge empfangen, so ist die Anzahl der Fehler $\operatorname{dist}(\mathbf{c}, \mathbf{y}) \geq 0$.

Definition 2.1.3 (Hamming-Gewicht). *Sei $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$ ein Vektor. Das* Hamming-Gewicht $\operatorname{wt}(\mathbf{v})$ *ist die Anzahl der Stellen von $\mathbf{v}$, die nicht null sind:* $\operatorname{wt}(\mathbf{v}) \coloneqq \operatorname{dist}(\mathbf{v}, 0)$.

Ein fehlerkorrigierender Code $\mathcal{C} \subseteq \mathbb{F}_q^n$ kann t Fehler korrigieren, wenn für jeden Vektor $\mathbf{y} \in \mathbb{F}_q^n$ gilt, dass es höchstens ein Codewort $\mathbf{c} \in \mathcal{C}$ gibt, so dass $\text{dist}(\mathbf{y}, \mathbf{c}) \leq t$ gilt [23, S. 160].

Definition 2.1.4 (Fehlerkorrigierende Decodierer). *Sei $t \in \mathbb{N}$. Ein* fehlerkorrigierender Decodierer *$D_\mathcal{C}$ für fehlerbehaftete Codewörter eines Codes $\mathcal{C}$ ist eine Abbildung $D_\mathcal{C} \colon \mathbb{F}_q^n \to \mathcal{C}$ und kann t Fehler beheben, falls für alle Codewörter $\mathbf{c} \in \mathcal{C}$ und alle Fehlervektoren $\mathbf{e} \in \mathbb{F}_q^n$ mit $\text{wt}(\mathbf{e}) \leq t$ gilt: $D_\mathcal{C}(\mathbf{c} + \mathbf{e}) = \mathbf{c}$.*

Aus Definition 2.1.4 folgt, dass ein fehlerkorrigierender Decodierer, der t Fehler korrigieren kann, stets auch weniger als t Fehler korrigieren kann. Ein Beispiel für eine solche Abbildung ist der Patterson-Algorithmus (Algorithmus 2.3.1), der in Abschnitt 2.3 vorgestellt wird.

2.2 Lineare Codes

Lineare Codes sind eine besondere Klasse fehlerkorrigierender Codes und besitzen algebraische Eigenschaften, die ihre Verwendung für die in dieser Arbeit vorgestellten Verfahren besonders vorteilhaft machen.

Definition 2.2.1 (Lineare Codes). *Ein Code $\mathcal{C} \subseteq \mathbb{F}_q^n$ heißt* (n, k)-linearer Code *genau dann, wenn $\mathcal{C}$ ein Untervektorraum von $\mathbb{F}_q^n$ mit Dimension $\dim(\mathcal{C}) = k$ für ein $k < n$ ist.*

Da lineare Codes Vektorräume sind, lassen sie sich vollständig durch eine Basis beschreiben und erlauben schnelle Codierungs- und Decodierungsoperationen auf Grundlage ihrer Generatormatrix [19, S. 48]:

Definition 2.2.2 (Generatormatrix). *Eine* Generatormatrix *$G \in \mathbb{F}_q^{k \times n}$ eines (n, k)-linearen Codes $\mathcal{C}$ ist eine $k \times n$-Matrix, deren Zeilen eine Basis für $\mathcal{C}$ bilden.*

Offensichtlich gilt $\mathcal{C} = \{ \mathbf{v} \cdot G \mid \mathbf{v} \in \mathbb{F}_q^k \}$, wobei $G \in \mathbb{F}_q^{k \times n}$ eine Generatormatrix für den (n, k)-linearen Code $\mathcal{C}$ ist. Folglich lassen sich Nachrichten $\mathbf{v} \in \mathbb{F}_q^k$ im Fall linearer Codes über das Vektor-Matrix-Produkt $\mathbf{v} \cdot G$ als Codewörter $\mathbf{c} \in \mathcal{C}$ codieren. Umgekehrt ist das Decodieren eines Codewortes $\mathbf{c} \in \mathcal{C}$ zu einer Nachricht

$\mathbf{v}$ äquivalent dazu, das lineare Gleichungssystem $G^\intercal \cdot \mathbf{v} = \mathbf{c}$ aus n Gleichungen zu lösen. Die Dimension des Zeilenraums von $G^\intercal$ entspricht der Dimension k von $\mathcal{C}$ und somit der Anzahl der Unbekannten, also existiert eine eindeutige Lösung des Gleichungssystems für alle $\mathbf{c} \in \mathcal{C}$ und keine Lösung für $\mathbf{c} \notin \mathcal{C}$.

Definition 2.2.3 (Dualer Code). *Der* duale Code $\mathcal{C}^\perp$ *eines* (n,k)*-linearen Codes* $\mathcal{C} \subseteq \mathbb{F}_q^n$ *ist die Menge der Vektoren, die orthogonal zu* $\mathcal{C}$ *sind:*

$$\mathcal{C}^\perp \coloneqq \{\mathbf{x} \in \mathbb{F}_q^n \mid \forall \mathbf{c} \in \mathcal{C} \colon \langle \mathbf{x}, \mathbf{c} \rangle = 0\}$$

Da $\mathcal{C}^\perp$ ein Untervektorraum von $\mathbb{F}_q^n$ und ein Komplementärraum von $\mathcal{C}$ ist, gilt für die Dimension des dualen Codes $\dim(\mathcal{C}^\perp) = \dim(\mathbb{F}_q^n) - \dim(\mathcal{C}) = n - k$, also ist $\mathcal{C}^\perp$ ein $(n, n-k)$-linearer Code.

Definition 2.2.4 (Kontrollmatrix). *Eine* Kontrollmatrix $H \in \mathbb{F}_q^{(n-k)\times n}$ *eines* (n,k)*-linearen Codes* $\mathcal{C}$ *ist eine Generatormatrix des dualen Codes* $\mathcal{C}^\perp$*.*

Jede Generatormatrix eines linearen Codes $\mathcal{C}$ spannt durch ihren Zeilenraum denselben Code auf, aber die Abbildung von Nachrichten auf Codewörter ist nicht dieselbe. Ebenso hängt das Syndrom von der verwendeten Kontrollmatrix ab:

Definition 2.2.5 (Syndrom). *Das* Syndrom *eines Vektors* $\mathbf{y} \in \mathbb{F}_q^n$ *bezüglich einer Kontrollmatrix* H *eines* (n,k)*-linearen Codes* $\mathcal{C}$ *ist das Matrix-Vektor-Produkt* $H \cdot \mathbf{y}$*.*

Aus Definition 2.2.4 und 2.2.5 lässt sich folgern, dass für eine beliebige Generatormatrix G und eine beliebige Kontrollmatrix H eines linearen Codes $\mathcal{C}$ stets $G \cdot H^\intercal = 0$ gilt. Außerdem ist das Syndrom eines Vektors $\mathbf{c}$ bezüglich H genau dann 0, wenn $\mathbf{c}$ ein Codewort ist:

$$H \cdot \mathbf{c} = 0 \iff \mathbf{c} \in \mathcal{C}$$

Aus dieser Eigenschaft folgt, dass das Syndrom einer Nachricht $\mathbf{y} = \mathbf{c} + \mathbf{e}$ nur vom Fehlervektor $\mathbf{e}$ und nicht vom übertragenen Codewort $\mathbf{c} \in \mathcal{C}$ abhängt:

$$H \cdot \mathbf{y} = H \cdot (\mathbf{c} + \mathbf{e}) = H \cdot \mathbf{c} + H \cdot \mathbf{e} = H \cdot \mathbf{e}$$

Aus diesem Grund wird das Syndrom häufig als Grundlage für die Fehlerkorrektur linearer Codes verwendet. Ein einfaches Beispiel ist die Einteilung von Fehlervektoren in Klassen anhand ihres jeweiligen Syndroms, um die Brute-Force-Suche nach dem tatsächlichen Fehlervektor zu beschleunigen.

Definition 2.2.6 (Minimale Distanz). *Die* minimale Distanz d *eines* (n, k)*-linearen Codes ist die minimale Hamming-Distanz zweier Codewörter:*

$$d = \min\{\, \mathrm{dist}(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y} \,\}$$

Wir bezeichnen einen solchen linearen Code dann als (n, k, d)*-linearen Code.*

Ein (n, k, d)-linearer Code $\mathcal{C}$ kann allgemein r Fehler korrigieren, wenn $2r+1 \leq d$ ist, da in diesem Fall für jede Nachricht $\mathbf{y} \in \mathbb{F}_q^n$ maximal ein Codewort $\mathbf{c} \in \mathcal{C}$ mit $\mathrm{dist}(\mathbf{y}, \mathbf{c}) \leq r$ existiert. Wurde ein Codewort mit mehr Fehlern übertragen, ist eine eindeutige Fehlerkorrektur oft nicht korrekt möglich.

2.3 Binäre Goppa-Codes

Goppa-Codes sind eine besondere Teilmenge linearer Codes, die zuerst von Valery Denisovich Goppa beschrieben wurden. Für kryptographische Anwendungen spielen insbesondere binäre Goppa-Codes eine Rolle, welche wiederum einen Spezialfall allgemeiner Goppa-Codes darstellen.

Im Folgenden seien $t, m, n \in \mathbb{N}$, $g(x) = g_0 + g_1 x + \cdots + g_t x^t \in \mathbb{F}_{2^m}[x]$ ein Polynom vom Grad t und $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{2^m}^n$ ein Vektor paarweise verschiedener Elemente aus $\mathbb{F}_{2^m}$ mit $g(\alpha_i) \neq 0$ für $1 \leq i \leq n$.

Definition 2.3.1 (Goppa-Syndrom). *Das* Goppa-Syndrom $S_{\mathbf{y}} \colon \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ *eines binären Goppa-Codes* $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ *für einen Vektor* $\mathbf{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_2^n$ *ist*

$$S_{\mathbf{y}}(x) = \sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} \in \mathbb{F}_{2^m}[x],$$

wobei $\frac{1}{x-\alpha_i}$ *das eindeutige Polynom mit* $\frac{1}{x-\alpha_i} \cdot (x - \alpha_i) \equiv 1 \pmod{g(x)}$ *ist.*

Definition 2.3.2 (Binäre Goppa-Codes). *Ein* binärer Goppa-Code $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ *ist die Menge*

$$\Gamma(\boldsymbol{\alpha}, g) = \{\, \mathbf{c} \in \mathbb{F}_2^n \mid S_{\mathbf{c}}(x) \equiv 0 \pmod{g(x)} \,\}, \tag{2.1}$$

also die Menge aller Vektoren $\mathbf{c} \in \mathbb{F}_2^n$*, für die* $g(x)$ *das Polynom* $S_{\mathbf{c}}(x)$ *restlos teilt.*

Ein binärer Goppa-Code $\mathcal{C}$ mit obigen Parametern ist ein (n, k, d)-linearer Code, dabei gilt für die Dimension $\dim(\mathcal{C}) = k \geq n - mt$ (Beweisskizze anhand der Kontrollmatrix folgt). Die minimale Distanz d von $\mathcal{C}$ ist stets mindestens $t + 1$ [15, S. 3f.].

Eine Kontrollmatrix H für $\mathcal{C}$ lässt sich auf verschiedene Arten konstruieren, hier soll sie aus der Definition des Goppa-Syndroms hergeleitet werden. Dazu wird das Goppa-Syndrom $S_{\mathbf{c}}(x)$ für einen Vektor $\mathbf{c} \in \mathbb{F}_2^n$ als Summe dargestellt, welche unmittelbar aus Definition 2.3.1 folgt:

$$S_{\mathbf{c}}(x) = \sum_{i=1}^{n} c_i p_i(x) \quad \text{mit } p_i(x) = \frac{1}{x - \alpha_i}$$

Laut obiger Definition ist $p_i(x)$ eindeutig bestimmt mit Grad kleiner als t und kann dargestellt werden als $p_i(x) = p_{i1} + p_{i2}x + \cdots + p_{it}x^{t-1}$. Da also der Grad von $S_{\mathbf{c}}(x)$ auch kleiner als t ist und somit kleiner als der Grad von $g(x)$, wird die Kongruenz $S_{\mathbf{c}}(x) \equiv 0 \pmod{g(x)}$ nur von $S_{\mathbf{c}}(x) = 0$ erfüllt, was genau dann der Fall ist, wenn jeder Koeffizient von $S_{\mathbf{c}}(x)$ null ist. Der Koeffizient von x^j in $S_{\mathbf{c}}(x)$ ist $\sum_{i=1}^{n} c_i p_{ij}$. Ausgehend von Gleichung 2.1 folgern wir

$$\mathbf{c} \in \mathcal{C} \iff \sum_{i=1}^{n} c_i p_{ij} = 0 \text{ für } 1 \leq j \leq t.$$

Diese Bedingung lässt sich durch ein lineares Gleichungssystem über $\mathbb{F}_{2^m}$ mit t Gleichungen darstellen, die Faktoren des Gleichungssystems bilden dann die vorläufige Kontrollmatrix H' über $\mathbb{F}_{2^m}$:

$$H' = \begin{pmatrix} p_{11} & \cdots & p_{n1} \\ \vdots & \ddots & \vdots \\ p_{1t} & \cdots & p_{nt} \end{pmatrix}$$

Für die Werte der Faktoren gilt nach Jochemsz [15]

$$p_{ij} = \frac{\sum_{l=j}^{t} g_l \alpha_i^{l-j}}{g(\alpha_i)},$$

was dem Produkt $H' = C \cdot X \cdot Y$ mit

$$C = \begin{pmatrix} g_t & g_{t-1} & g_{t-2} & \dots & g_1 \\ 0 & g_t & g_{t-1} & \dots & g_2 \\ 0 & 0 & g_t & \dots & g_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & g_t \end{pmatrix}, \quad X = \begin{pmatrix} \alpha_1^{t-1} & \dots & \alpha_n^{t-1} \\ \alpha_1^{t-2} & \dots & \alpha_n^{t-2} \\ \vdots & \ddots & \vdots \\ \alpha_1 & \dots & \alpha_n \\ 1 & \dots & 1 \end{pmatrix},$$

$$Y = \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & \dots & 0 \\ 0 & \frac{1}{g(\alpha_2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{g(\alpha_n)} \end{pmatrix}$$

entspricht.

Die Reihenfolge der Zeilen von H' spielt dabei keine Rolle. Da C invertierbar ist, gilt $\ker(C \cdot X \cdot Y) = \ker(X \cdot Y)$, also kann auch $X \cdot Y$ als Grundlage für eine Kontrollmatrix verwendet werden, die denselben Code erzeugt.

Die Matrix $H' \in \mathbb{F}_{2^m}^{t \times n}$ muss schließlich noch in eine Kontrollmatrix H über $\mathbb{F}_2$ umgewandelt werden. Dazu wird jede Zahl aus $\mathbb{F}_{2^m}$ durch einen Spaltenvektor in $\mathbb{F}_2^m$ ersetzt (siehe Anhang A). Das Ergebnis ist $H \in \mathbb{F}_2^{mt \times n}$. Aus dieser Umwandlung folgt zugleich auch die oben genannte Dimension $k \geq n - mt$ von $\mathcal{C}$, da die Dimension von $\mathcal{C}^\perp$ der Dimension des Zeilenraums von H entspricht und somit maximal mt sein kann.

Aus der Kontrollmatrix H kann eine Generatormatrix G hergeleitet werden. Dazu kann H durch elementare Zeilenoperationen in eine systematische Form gebracht werden, in der die ersten mt Spalten die Einheitsmatrix bilden: $H_{\text{sys}} = \begin{pmatrix} I_{mt} & P \end{pmatrix}$. Dann ist $G = \begin{pmatrix} P^\intercal & I_k \end{pmatrix}$ eine Generatormatrix für den Code [19, S. 49ff.].

Definition 2.3.3 (Separable Polynome). *Ein Polynom $p \in \mathbb{F}_q[x]$ heißt* separabel, *wenn alle Nullstellen von p einfach sind.*

Ist g separabel, so ist die minimale Distanz d von $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ sogar mindestens $2t + 1$. Solche (separablen) binären Goppa-Codes können also t Fehler korrigieren [15, S. 4f.].

Definition 2.3.4 (Irreduzible Polynome). *Ein Polynom $p \in \mathbb{F}_q[x]$ heißt* irreduzibel *über $\mathbb{F}_q$, wenn der Grad von p positiv ist und für jede Faktorisierung $p = ab$ für $a, b \in \mathbb{F}_q[x]$ gilt, dass a oder b ein konstantes Polynom ist [19, S. 23].*

Mit dieser Definition können wir eine weitere Teilklasse binärer Goppa-Codes einführen:

Definition 2.3.5 (Irreduzible binäre Goppa-Codes). *Ein binärer Goppa-Code $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ heißt* irreduzibel *genau dann, wenn $g(x)$ irreduzibel ist [19, S. 196].*

Zusätzlich zu dieser Definition benötigen wir eine wichtige Feststellung und eine sich ergebende Folgerung:

Satz 2.3.1 ([13, S. 549]). *Jedes irreduzible Polynom p über einem endlichen Körper $\mathbb{F}_q$ ist separabel.*

Beweis. Sei p irreduzibel, aber nicht separabel. Dann existiert eine (mehrfache) Nullstelle x_0 von p und somit ist $(x - x_0)$ ein nicht-trivialer Teiler von p, also ist p nicht irreduzibel laut Definition 2.3.4. Widerspruch. □

Korollar 2.3.1. *Das Polynom g eines irreduziblen binären Goppa-Codes $\Gamma(\boldsymbol{\alpha}, g)$ ist separabel, also ist die minimale Distanz d von $\Gamma(\boldsymbol{\alpha}, g)$ mindestens $2t + 1$.*

Im Gegensatz zu zufälligen linearen Codes existiert für jeden binären Goppa-Code $\mathcal{C}$ ein effizienter fehlerkorrigierender Decodierer $D_{\mathcal{C}}$ (siehe Definition 2.1.4), also ein Algorithmus, der in polynomieller Zeit Übertragungsfehler korrigiert. Am bekanntesten ist der Algorithmus von Patterson, der im Fall von irreduziblen binären Goppa-Codes t Fehler eindeutig korrigieren kann. Die folgende Darstellung fasst die Erläuterungen von Jochemsz in [15, S. 17–21] und Overbeck et al. in [25, S. 139f.] zusammen und ergänzt sie um zusätzliche Erklärungen.

Im Folgenden sei $\mathbf{y} \in \mathbb{F}_2^k$ ein fehlerbehaftetes Codewort mit $r \leq t$ Fehlern, also $\mathbf{y} = \mathbf{c} + \mathbf{e}$ für ein Codewort $\mathbf{c} \in \Gamma(\boldsymbol{\alpha}, g)$ und einen Fehlervektor $\mathbf{e} \in \mathbb{F}_2^k$ mit $\mathrm{wt}(\mathbf{e}) = r$.

Gesucht sind die Indizes $B = \{ i \mid e_i \neq 0 \}$, an denen die Fehler aufgetreten sind. Da der Code binär ist, ist dies ausreichend, um den Vektor $\mathbf{e}$ zu rekonstruieren; für nicht-binäre Goppa-Codes müssen zudem auch die Werte e_i an den Stellen $i \in B$ ermittelt werden.

Wir definieren das sogenannte *error locator polynomial* als

$$\sigma(x) = \prod_{i \in B} (x - \alpha_i). \tag{2.2}$$

Nach dieser Definition folgt B aus den Nullstellen von $\sigma(x)$ mit $|B| = \deg(\sigma) = r$. Als nächstes stellen wir fest, dass das Goppa-Syndrom von $\mathbf{y}$ dem von $\mathbf{e}$ entspricht, also nicht von $\mathbf{c}$ abhängt:

$$\begin{aligned} S_{\mathbf{y}}(x) &= \sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} \\ &= \sum_{i=1}^{n} \frac{c_i + e_i}{x - \alpha_i} \\ &= S_{\mathbf{c}}(x) + S_{\mathbf{e}}(x) \\ &\equiv S_{\mathbf{e}}(x) \pmod{g(x)} \end{aligned}$$

Damit gilt nun auch

$$\begin{aligned} \sigma(x) S_{\mathbf{y}}(x) &\equiv \sigma(x) S_{\mathbf{e}}(x) \pmod{g(x)} \\ &= \left(\prod_{i \in B} (x - \alpha_i) \right) \cdot \left(\sum_{i=1}^{n} \frac{e_i}{x - \alpha_i} \right) \\ &= \sum_{i \in B} e_i \prod_{j \in B \setminus \{i\}} (x - \alpha_j) \\ &= \sum_{i \in B} \prod_{j \in B \setminus \{i\}} (x - \alpha_j) \\ &= \sigma'(x). \end{aligned}$$

Es ist also ausreichend, die nachfolgende Gleichung zu lösen, um σ zu ermitteln:

$$\sigma(x) S_{\mathbf{e}}(x) \equiv \sigma'(x) \pmod{g(x)} \tag{2.3}$$

Aus Gleichung 2.2 folgt, dass $\sigma(x)$ ein Polynom mit $\deg(\sigma) = r \leq t$ ist. Wir teilen $\sigma(x)$ auf in zwei Polynome $a(x) = a_0 + a_1 x + \cdots + a_k x^k$ mit Grad $k \leq \frac{t}{2}$ und $b(x) = b_0 + b_1 x + \cdots + b_\ell x^\ell$ mit Grad $\ell \leq \frac{t-1}{2}$.

$$\begin{aligned}\sigma(x) &= a^2(x) + b^2(x)x \\ &= a_0^2 + a_1^2 x^2 + \cdots + a_k^2 z^{2k} + b_0^2 z + b_1^2 z^3 + \cdots + b_\ell^2 z^{2\ell+1}\end{aligned}$$

Diese Aufteilung funktioniert, da andere Summanden wie beispielsweise $a_0 a_1 x$ stets genau zweifach auftreten und Addition und Subtraktion in $\mathbb{F}_{2^m}$ dieselbe Operation sind, also verändert die zweifache Addition dieser Summanden das Ergebnis nicht.

Die Ableitung von σ ist nun

$$\begin{aligned}\sigma'(x) &= a(x)a'(x) + a(x)a'(x) + b(x)b'(x) + b(x)b'(x) + b^2(x) \\ &= a(x)a'(x) - a(x)a'(x) + b(x)b'(x) - b(x)b'(x) + b^2(x) \\ &= b^2(x).\end{aligned}$$

Eingesetzt in Gleichung 2.3 folgt

$$(a^2(x) + b^2(x)x)S_{\mathbf{e}}(x) \equiv b^2(x) \pmod{g(x)}. \tag{2.4}$$

Da $g(x)$ irreduzibel ist, sind $g(x)$ und $S_{\mathbf{e}}(x)$ teilerfremd (relativ prim) und der erweiterte euklidische Algorithmus liefert ein Polynom $h(x)$ mit

$$h(x)S_{\mathbf{e}}(x) \equiv 1 \pmod{g(x)}. \tag{2.5}$$

Gilt $h(x) = x$, setze $\sigma(x) = h(x)$. Offensichtlich ist nun $\sigma'(x) = 1$ und damit Gleichung 2.3 äquivalent zu Gleichung 2.5, also ist $\sigma(x) = x$ tatsächlich das gesuchte Polynom. Andernfalls multiplizieren wir beide Seiten von Gleichung 2.4 mit $h(x)$ und erhalten

$$a^2(x) + b^2(x)x \equiv b^2(x)h(x) \pmod{g(x)}.$$

Addition von $b^2(x)x$ zu beiden Seiten führt zu

$$a^2(x) \equiv b^2(x)(h(x) + x) \pmod{g(x)}. \tag{2.6}$$

Da $h(x) \neq x$, ist $h(x) + x \neq 0$. Außerdem existiert ein eindeutiges Polynom $d(x)$ mit $d^2(x) \equiv z + h(z) \pmod{g(x)}$. Einsetzen in Gleichung 2.6 liefert

$$a^2(x) \equiv b^2(x)d^2(x) \pmod{g(x)},$$

was wiederum äquivalent zu

$$a(x) \equiv b(x)d(x) \pmod{g(x)}$$

ist. Die Polynome $a(x)$ und $b(x)$ können nun mit dem erweiterten euklidischen Algorithmus bestimmt werden und das gesuchte Polynom ist $\sigma(x) = a^2(x) + b^2(x)x$.

Schließlich wird die Menge B anhand der Nullstellen von $\sigma(x)$ ermittelt und auf diese Weise der Fehlervektor $\mathbf{e}$ rekonstruiert. Das übertragene Codewort ist dann $\mathbf{c} = \mathbf{y} - \mathbf{e} \in \Gamma(\boldsymbol{\alpha}, g)$. Diese Überlegungen sind in Algorithmus 2.3.1 zusammengefasst.

Algoritmus 2.3.1 Patterson-Algorithmus für irreduzible binäre Goppa-Codes

Eingabe: Nachricht $\mathbf{y} \in \mathbb{F}_2^n$, Goppa-Code $\Gamma(\boldsymbol{\alpha}, g)$
Ausgabe: Fehlervektor $\mathbf{e} \in \mathbb{F}_2^n$

1: $s(x) \leftarrow S_{\mathbf{y}}(x)$ ▷ Syndrom berechnen
2: **if** $s(x) = 0$ **then** ▷ $\mathbf{y}$ ist ein Codewort
3: $\quad \mathbf{e} \leftarrow 0$
4: **else** ▷ $\mathbf{y}$ ist fehlerbehaftet
5: $\quad h(x) \leftarrow (s(x))^{-1} \pmod{g(x)}$ ▷ Löse $h(x)s(x) \equiv 1$ mit EEA
6: $\quad$ **if** $h(x) = x$ **then**
7: $\quad\quad \sigma(x) \leftarrow h(x)$
8: $\quad$ **else**
9: $\quad\quad d(x) \leftarrow \sqrt{h(x) + x} \pmod{g(x)}$
10: $\quad\quad$ Finde a, b mit $d(x)b(x) \equiv a(x) \pmod{g(x)}$ und $\deg(a) \leq \frac{t}{2}$, $\deg(b) \leq \frac{t-1}{2}$.
11: $\quad\quad \sigma(x) \leftarrow a^2(x) + b^2(x)x$
12: $\quad B \leftarrow \{\, i \mid \sigma(\alpha_i) = 0 \,\}$ ▷ Finde fehlerbehaftete Stellen
13: $\quad$ **for** i **from** 1 **to** n **do** ▷ Rekonstruiere Fehlervektor $\mathbf{e}$
14: $\quad\quad$ **if** $i \in B$ **then** $e_i \leftarrow 1$ **else** $e_i \leftarrow 0$

2.4 Komplexitätstheoretische Probleme

In diesem Abschnitt werden zwei grundlegende komplexitätstheoretische Probleme definiert, auf denen die Sicherheit der in den folgenden Kapiteln vorgestellten codebasierten Verfahren basiert. Das erste Problem drückt die Schwierigkeit aus, aus einem fehlerbehafteten Codewort basierend auf einer Generatormatrix das übertragene Codewort zu ermitteln.

Definition 2.4.1 (General Decoding-Problem). *Sei $\mathbb{F}_q$ ein endlicher Körper, $\mathcal{C} \subseteq \mathbb{F}_q^n$ ein (n,k)-linearer Code mit Generatormatrix $G \in \mathbb{F}_q^{k \times n}$, $\mathbf{c} \in \mathbb{F}_q^n$ ein Vektor und $t \in \mathbb{N}$. Das* General Decoding-Problem (GD-Problem) *besteht darin, festzustellen, ob ein $\mathbf{m} \in \mathbb{F}_q^k$ existiert, so dass $\mathrm{wt}(\mathbf{e}) \leq t$ für $\mathbf{e} = \mathbf{c} - \mathbf{m} \cdot G$ gilt.*

Das zweite Problem drückt die Schwierigkeit aus, von dem Syndrom einer Nachricht ausgehend von einer Kontrollmatrix auf den Fehlervektor zu schließen.

Definition 2.4.2 (Syndrome Decoding-Problem). *Sei $\mathbb{F}_q$ ein endlicher Körper, $\mathcal{C} \subseteq \mathbb{F}_q^n$ ein (n,k)-linearer Code mit Kontrollmatrix $H \in \mathbb{F}_q^{(n-k) \times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ ein Vektor und $t \in \mathbb{N}$. Das* Syndrome Decoding-Problem (SD-Problem) *besteht darin, festzustellen, ob ein $\mathbf{e} \in \mathbb{F}_q^n$ mit $\mathrm{wt}(\mathbf{e}) \leq t$ exisiert, so dass $H \cdot \mathbf{e} = \mathbf{s}$ gilt.*

Die beiden folgenden Sätze zeigen, dass die Probleme gleichermaßen schwer sind:

Satz 2.4.1. *Das* GD*-Problem ist in Polynomialzeit auf das* SD*-Problem reduzierbar.*

Beweis. Gegeben sei eine Instanz $I_{\mathsf{GD}} = (\mathbb{F}_q, \mathcal{C}, G, \mathbf{c}, t)$ des General Decoding-Problems. Sei H eine Kontrollmatrix für $\mathcal{C}$ mit $G \cdot H^\intercal = 0$ und $\mathbf{s} = H \cdot \mathbf{c}$ das Syndrom von $\mathbf{c}$. Dann ist $I_{\mathsf{GD}} \in \mathsf{GD}$ genau dann, wenn $I_{\mathsf{SD}} = (\mathbb{F}_q, \mathcal{C}, H, \mathbf{s}, t) \in \mathsf{SD}$ gilt:

Angenommen, $I_{\mathsf{GD}} \in \mathsf{GD}$, dann existiert ein $\mathbf{m}$ mit $\mathrm{wt}(\mathbf{e}) \leq t$ für $\mathbf{e} = \mathbf{c} - \mathbf{m} \cdot G$. Multiplikation mit H von links führt zu $H \cdot \mathbf{e} = H \cdot (\mathbf{c} - \mathbf{m} \cdot G) = H \cdot \mathbf{c} = \mathbf{s}$, also ist $\mathbf{e}$ eine Lösung von I_{SD}, folglich gilt $I_{\mathsf{SD}} \in \mathsf{SD}$.

Gilt umgekehrt $I_{\mathsf{GD}} \notin \mathsf{GD}$, so existiert kein solches $\mathbf{e}$, da ansonsten $H \cdot (\mathbf{c} - \mathbf{e}) = 0$ gelten würde und somit $\mathbf{c} - \mathbf{e}$ ein Codewort wäre, dann würde allerdings ein $\mathbf{m}$ mit $\mathbf{c} - \mathbf{e} = \mathbf{m} \cdot G$ existieren und somit $I_{\mathsf{GD}} \in \mathsf{GD}$ gelten. Also gilt $I_{\mathsf{SD}} \notin \mathsf{SD}$. □

Satz 2.4.2. *Das* SD*-Problem ist in Polynomialzeit auf das* GD*-Problem reduzierbar.*

Beweis. Gegeben sei eine Instanz $I_{\mathsf{SD}} = (\mathbb{F}_q, \mathcal{C}, H, \mathbf{s}, t)$ des Syndrome Decoding-Problems. Sei G eine Kontrollmatrix für $\mathcal{C}$ mit $G \cdot H^\intercal = 0$. Finde einen beliebigen Vektor $\mathbf{c}$ mit $H \cdot \mathbf{c} = \mathbf{s}$. (Dies ist in Polynomialzeit möglich.) Dann ist $I_{\mathsf{SD}} \in \mathsf{SD}$ genau dann, wenn $I_{\mathsf{GD}} = (\mathbb{F}_q, \mathcal{C}, G, \mathbf{c}, t) \in \mathsf{GD}$ gilt:

Angenommen, $I_{\mathsf{SD}} \in \mathsf{SD}$, dann existiert ein $\mathbf{e}$ mit $H \cdot \mathbf{e} = \mathbf{s}$ und $\mathrm{wt}(\mathbf{e}) \leq t$. Dann gilt auch $H \cdot \mathbf{e} = \mathbf{s} = H \cdot \mathbf{c}$, also $H \cdot (\mathbf{c} - \mathbf{e}) = 0$, also ist $\mathbf{c} - \mathbf{e}$ ein Codewort, also existiert ein $\mathbf{m}$ mit $\mathbf{m} \cdot G = \mathbf{c} - \mathbf{e} \iff \mathbf{c} = \mathbf{m} \cdot G + \mathbf{e}$, also $I_{\mathsf{GD}} \in \mathsf{GD}$.

Gilt umgekehrt $I_{\mathsf{SD}} \notin \mathsf{SD}$, so existiert kein $\mathbf{e}$ mit $H \cdot \mathbf{e} = \mathbf{s}$ und $\mathrm{wt}(\mathbf{e}) \leq t$. Wäre $I_{\mathsf{GD}} \in \mathsf{GD}$, so würde $\mathbf{c} = \mathbf{m} \cdot G + \mathbf{e}$ für ein beliebiges $\mathbf{m}$ und ein $\mathbf{e}$ mit $\mathrm{wt}(\mathbf{e}) \leq t$ gelten. Multiplikation mit H von links liefert $\mathbf{s} = H \cdot \mathbf{e}$, also wäre $\mathbf{e}$ eine Lösung für I_{SD} und somit $I_{\mathsf{SD}} \in \mathsf{SD}$. □

Tatsächlich sind beide Probleme NP-vollständig [25, S. 107] und können somit von klassischen Computern im Allgemeinen nicht effizient gelöst werden, es sei denn, $\mathsf{P} = \mathsf{NP}$ gilt, was ein ungelöstes Problem der Komplexitätstheorie darstellt. Die Frage, ob NP-vollständige Probleme von Quantencomputern effizient gelöst werden können, ist bislang ebenso wenig eindeutig beantwortet worden. Grover hat gezeigt, dass Quantenberechnungen die Zeit zur Entscheidung eines Problems aus NP auf die Quadratwurzel der klassischerweise benötigten Zeit reduzieren können, also auf $\mathcal{O}(2^{n/2})$ statt $\mathcal{O}(2^n)$, was jedoch nach wie vor eine exponentielle Laufzeit bedeutet. Unter der Voraussetzung $\mathsf{P} \neq \mathsf{NP}$ gilt es als unwahrscheinlich, dass Quantencomputer NP-vollständige Probleme in Polynomialzeit lösen können [1, 8].

3 Codebasierte Verfahren

In diesem Kapitel sollen die bekanntesten codebasierten Verfahren, die auf den in Kapitel 2 erläuterten Grundlagen zur Codierungstheorie beruhen, vorgestellt werden. Wir vernachlässigen an dieser Stelle symmetrische Kryptographie, da etablierte symmetrische Verfahren wie AES nach aktuellen Kenntnissen im Gegensatz zu verbreiteten asymmetrischen Verfahren wie RSA und Diffie-Hellman auch gegen Quantencomputer ausreichend Schutz bieten [4, S. 2].

Mit asymmetrischen Verschlüsselungsverfahren bezeichnen wir im Folgenden wie üblich Systeme, die Mechanismen zur Schlüsselerzeugung, Verschlüsselung und Entschlüsselung definieren (siehe Abbildung 1). Ver- und Entschlüsselung verwenden dabei verschiedene Schlüssel: Üblicherweise soll jeder Kommunikationspartner (Sender) in der Lage sein, Nachrichten zu verschlüsseln, aber nur der Empfänger soll die Nachrichten entschlüsseln können. Der zur Verschlüsselung verwendete Schlüssel ist somit in der Regel allen Teilnehmern bekannt und wird als *öffentlicher Schlüssel* bezeichnet. Der *private Schlüssel* ist dagegen nur dem Empfänger bekannt und dient diesem zur Entschlüsselung. Öffentlicher und privater Schlüssel zusammen werden als Schlüsselpaar bezeichnet.

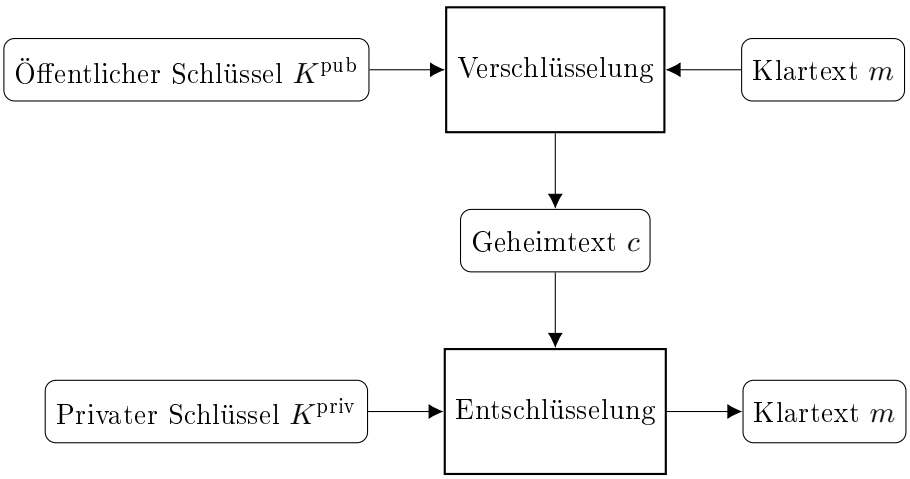

Abbildung 1: Struktur eines asymmetrischen Verschlüsselungsschemas

3.1 McEliece

Das McEliece-Verfahren wurde nach seinem Erfinder, Robert McEliece, benannt, der es 1978 vorstellte [21]. Es handelt sich bei diesem Verfahren um ein asymmetrisches Verschlüsselungsschema. Die Grundidee ist, die Nachricht als Codewort zu codieren und zufällig Fehler hinzuzufügen, die ausschließlich der Besitzer des privaten Schlüssels korrigieren kann, indem er die nur ihm bekannte zugrundeliegende Struktur des verwendeten Codes ausnutzt. McEliece schlug irreduzible binäre Goppa-Codes (siehe Abschnitt 2.3) vor, was nach wie vor als sicher gilt.

Um k-Bit-Nachrichten zu verschlüsseln, werden diese mit einem (n, k)-linearen Code $\mathcal{C}$ über $\mathbb{F}_2$, der t Fehler korrigieren kann, in n-Bit-Codewörter codiert. Die übertragenen Nachrichten sind also $\frac{n}{k}$-mal so lang wie der eigentliche Inhalt. Der Sender fügt anschließend t Fehler an zufälligen Stellen des Codewortes hinzu, was das Decodieren ohne Kenntnis über die Struktur des verwendeten Codes bei geeigneter Wahl der Parameter hinreichend schwierig macht:

Definition 3.1.1 (McEliece-Problem). *Sei* $(G^{pub}, t) \in \left(\mathbb{F}_2^{k \times n} \times \mathbb{N}\right)$ *ein öffentlicher McEliece-Schlüssel für* $n, k \in \mathbb{N}$ *mit* $k < n$ *und* $\mathbf{c} \in \mathbb{F}_2^n$ *eine Nachricht. Das McEliece-Problem besteht darin, die Nachricht* $\mathbf{m} \in \mathbb{F}_2^k$ *mit* $\mathrm{wt}(\mathbf{m} \cdot G^{pub} - \mathbf{c}) = t$ *zu finden [25, S. 98].*

Dieses Problem ist reduzierbar auf das General-Decoding-Problem [21, S. 115], aber im Allgemeinen nicht umgekehrt, da das McEliece-Problem auf eine Teilmenge aller linearen Codes beschränkt ist. Aus diesem Grund ist das McEliece-Problem nicht bekanntermaßen NP-vollständig [25, S. 98]. Allerdings existiert bislang auch kein Ansatz für eine Lösung des Problems in Polynomialzeit und somit kein Beweis dafür, dass das Problem in P liegt und somit effizient lösbar ist.

Um keine Rückschlüsse auf die Struktur des Codes zu ermöglichen, wird die Generatormatrix $G \in \mathbb{F}_2^{k \times n}$ von links mit einer zufälligen invertierbaren Matrix $S \in \mathbb{F}_2^{k \times k}$ und von rechts mit einer zufälligen Permutationsmatrix $P \in \mathbb{F}_2^{n \times n}$ multipliziert, was die Eigenschaften des Codes nicht verändert, aber die Struktur hinreichend verbirgt [21]:

$$G^{\mathrm{pub}} = S \cdot G \cdot P$$

Der öffentliche Schlüssel $K^{\text{pub}} = (G^{\text{pub}}, t)$ besteht aus der modifizierten Generatormatrix G^{pub} und der Anzahl der korrigierten Fehler t. Der private Schlüssel $K^{\text{priv}} = (S, D_{\mathcal{C}}, P)$ enthält die zur Entschlüsselung notwendigen Informationen, also die invertierbare Matrix S, die Permutationsmatrix P und einen effizienten fehlerkorrigierenden Decodierer $D_{\mathcal{C}}$ für $\mathcal{C}$, der t Fehler korrigieren kann (siehe Definition 2.1.4):

Algoritmus 3.1.1 McEliece — Schlüsselerzeugung

Ausgabe: Schlüsselpaar $(K^{\text{pub}}, K^{\text{priv}})$

1: Erzeuge einen zufälligen binären irreduziblen Goppa-Code $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ mit $\deg(g) = t$ und $|\boldsymbol{\alpha}| = n$ in $\mathbb{F}_{2^m}$.
2: Finde eine Generatormatrix $G \in \mathbb{F}_2^{k \times n}$ für $\mathcal{C}$.
3: Sei $D_{\mathcal{C}}$ ein effizienter fehlerkorrigierender Decodierer für $\mathcal{C}$, der t Fehler korrigieren kann.
4: Wähle eine zufällige invertierbare Matrix $S \in \mathbb{F}_2^{k \times k}$.
5: Wähle eine zufällige Permutationsmatrix $P \in \mathbb{F}_2^{n \times n}$.
6: $G^{\text{pub}} \leftarrow S \cdot G \cdot P$
7: $K^{\text{pub}} \leftarrow (G^{\text{pub}}, t)$
8: $K^{\text{priv}} \leftarrow (S, D_{\mathcal{G}}, P)$
9: **return** $(K^{\text{pub}}, K^{\text{priv}})$

Da das verwendete Polynom g des Codes $\mathcal{C} = \Gamma(\boldsymbol{\alpha}, g)$ irreduzibel ist, hat g keine Nullstellen in $\mathbb{F}_{2^m}$. Daher wird häufig $n = 2^m$ gesetzt, dann enthält $\boldsymbol{\alpha}$ alle Elemente des Körpers $\mathbb{F}_{2^m}$ [6, S. 69].

Zum Verschlüsseln einer Nachricht $\mathbf{m}$ wird diese in ein Codewort $\mathbf{c} \in \mathcal{C}$ anhand der Generatormatrix G^{pub} umgewandelt, bevor zufällig Fehler an t Positionen hinzugefügt werden:

Algoritmus 3.1.2 McEliece — Verschlüsselung

Eingabe: Klartext $\mathbf{m} \in \mathbb{F}_2^k$, öffentlicher Schlüssel $K^{\text{pub}} = (G^{\text{pub}}, t)$
Ausgabe: Geheimtext $\mathbf{c} \in \mathbb{F}_2^n$

1: Wähle einen zufälligen Vektor $\mathbf{e} \in \mathbb{F}_2^n$ mit $\text{wt}(\mathbf{e}) = t$.
2: $\mathbf{c} \leftarrow \mathbf{m} \cdot G^{\text{pub}} + \mathbf{e}$
3: **return c**

Zum Entschlüsseln einer Nachricht $\mathbf{c} = \mathbf{m} \cdot G^{\text{pub}} + \mathbf{e}$ müssen lediglich die Fehler korrigiert und der Einfluss der geheimen Matrizen S und P umgekehrt werden:

Algoritmus 3.1.3 McEliece — Entschlüsselung

Eingabe: Geheimtext $\mathbf{c} \in \mathbb{F}_2^n$, privater Schlüssel $K^{\text{priv}} = (S, D_{\mathcal{G}}, P)$
Ausgabe: Klartext $\mathbf{m} \in \mathbb{F}_2^k$

1: $\mathbf{c}_P \leftarrow \mathbf{c} \cdot P^{-1}$ ▷ Permutation umkehren
2: $\mathbf{c}' \leftarrow D_{\mathcal{C}}(\mathbf{c}_P)$ ▷ Fehler korrigieren: $\mathbf{c}' = \mathbf{m} \cdot S \cdot G$
3: Löse LGS $G^\intercal \cdot \mathbf{m}' = \mathbf{c}'$. ▷ Wandle Codewort $\mathbf{c}'$ in Nachricht $\mathbf{m}'$ um
4: $\mathbf{m} \leftarrow \mathbf{m}' \cdot S^{-1}$ ▷ Multiplikation mit S rückgängig machen
5: **return m**

McEliece weist durch die üblicherweise recht große Blockgröße und die einfachen Ver- und Entschlüsselungsalgorithmen hohe Übertragungsraten auf, allerdings sind im Vergleich zu etablierten asymmetrischen Verfahren wesentlich größere öffentliche Schlüssel notwendig, um äquivalente Sicherheit zu gewährleisten. Entsprechende Parameter werden in Abschnitt 4.2 diskutiert.

3.2 Niederreiter

Das von Harald Niederreiter 1986 vorgeschlagene Schema für asymmetrische Verschlüsselung ist McEliece sehr ähnlich. Anstatt die Nachricht in das Codewort zu codieren und dieses fehlerbehaftet zu übertragen, schlug er vor, den Klartext als Fehlervektor zu codieren und nur das Syndrom (siehe Definition 2.2.5) zu übertragen [23]. Dies hat den Nachteil, dass eine umkehrbare Abbildung $\Phi_{n,t}$ von der Menge der Klartexte $\mathbb{F}_2^{\mathcal{L}_{n,t}}$ mit Länge $\mathcal{L}_{n,t} = \lfloor \log_2 \binom{n}{t} \rfloor$ in die Menge $\mathcal{W}_{n,t} = \{\mathbf{e} \in \mathbb{F}_2^n \mid \text{wt}(\mathbf{e}) = t\}$ der Fehlervektoren mit Länge n und Gewicht t benötigt wird, was die Verschlüsselung potenziell verlangsamt. Die Umkehrabbildung $\Phi_{n,t}^{-1}$ wird analog zur Entschlüsselung verwendet. Eine effiziente Implementierung von $\Phi_{n,t}$ wird beispielsweise von Overbeck und Sendrier diskutiert [25].

Da zur Verschlüsselung kein Codewort, sondern das Syndrom berechnet werden muss, verwendet Niederreiter eine modifizierte Kontrollmatrix H^{pub} anstelle der von McEliece eingesetzten Generatormatrix für den Code, entsprechend ist der öffentliche Schlüssel $K^{\text{priv}} = (H^{\text{pub}}, t)$. Analog zu McEliece wird die Struktur dieser Matrix durch eine zufällige invertierbare Matrix $M \in \mathbb{F}_2^{(n-k)\times(n-k)}$ und eine zufällige

Permutationsmatrix $P \in \mathbb{F}_2^{n \times n}$ verborgen, so dass

$$H^{\text{pub}} = M \cdot H \cdot P$$

gilt. Der private Schlüssel ist dann $K^{\text{priv}} = (M, D_{\mathcal{C}}, P)$ mit einem Syndromdecodierungsalgorithmus $D_{\mathcal{C}}$ für $\mathcal{C}$ [23, S. 161]:

Definition 3.2.1. *Ein* Syndromdecodierungsalgorithmus *für eine Kontrollmatrix H eines linearen Codes $\mathcal{C}$ ist eine Funktion, die einem Syndrom $\mathbf{s} = H \cdot \mathbf{e}$ einen Fehlervektor $\mathbf{e}$ mit einem bestimmten (maximalem) Hamming-Gewicht zuordnet.*

Anmerkung. *Basierend auf den Reduktionen in Abschnitt 2.4 lässt sich aus jedem fehlerkorrigierenden Decodierer (Definition 2.1.4) ein Syndromdecodierungsalgorithmus konstruieren und umgekehrt.*

Algoritmus 3.2.1 Niederreiter — Schlüsselerzeugung

Ausgabe: Schlüsselpaar $(K^{\text{pub}}, K^{\text{priv}})$

1: Erzeuge einen zufälligen (n, k)-linearen Code $\mathcal{C}$, der t Fehler korrigieren kann.
2: Finde eine Kontrollmatrix $H \in \mathbb{F}_2^{(n-k) \times n}$ für $\mathcal{C}$.
3: Sei $D_{\mathcal{C}}$ ein effizienter Syndromdecodierungsalgorithmus für $\mathcal{C}$, der t Fehler korrigieren kann.
4: Wähle eine zufällige invertierbare Matrix $M \in \mathbb{F}_2^{(n-k) \times (n-k)}$.
5: Wähle eine zufällige Permutationsmatrix $P \in \mathbb{F}_2^{n \times n}$.
6: $H^{\text{pub}} \leftarrow M \cdot H \cdot P$
7: $K^{\text{pub}} \leftarrow (H^{\text{pub}}, t)$
8: $K^{\text{priv}} \leftarrow (P, D_{\mathcal{C}}, M)$
9: **return** $(K^{\text{pub}}, K^{\text{priv}})$

Zur Verschlüsselung muss die Nachricht $\mathbf{m} \in \mathbb{F}_2^k$ auf einen Fehlervektor $\mathbf{e} \in \mathbb{F}_2^n$ mit $\text{wt}(\mathbf{e}) = t$ abgebildet und das Syndrom des Fehlervektors anhand der modifizierten Kontrollmatrix H^{pub} bestimmt werden:

Algoritmus 3.2.2 Niederreiter — Verschlüsselung

Eingabe: Klartext $\mathbf{m} \in \mathbb{F}_2^{\mathcal{L}_{n,t}}$, öffentlicher Schlüssel $K^{\text{pub}} = (H^{\text{pub}}, t)$
Ausgabe: Geheimtext $\mathbf{s} \in \mathbb{F}_2^{n-k}$

1: $\mathbf{e} \leftarrow \Phi_{n,t}(\mathbf{m})$
2: $\mathbf{c} \leftarrow H^{\text{pub}} \cdot \mathbf{e}$
3: **return** $\mathbf{c}$

Um eine Nachricht $\mathbf{c} = H^{\text{pub}} \cdot \Phi_{n,t}(\mathbf{m})$ zu entschlüsseln, muss analog zu McEliece der Einfluss der Matrizen M und P umgekehrt werden und mithilfe von $D_{\mathcal{C}}$ vom Syndrom auf den Fehlervektor geschlossen werden:

Algoritmus 3.2.3 Niederreiter — Entschlüsselung

Eingabe: Geheimtext $\mathbf{s} \in \mathbb{F}_2^{n-k}$, privater Schlüssel $K^{\text{priv}} = (P, D_{\mathcal{C}}, M)$
Ausgabe: Klartext $\mathbf{m} \in \mathbb{F}_2^{\mathcal{L}_{n,t}}$

1: $\mathbf{s}' \leftarrow M^{-1} \cdot \mathbf{s}$ ▷ $\mathbf{s}' = H \cdot P \cdot \mathbf{e}$
2: $\mathbf{s} \leftarrow D_{\mathcal{C}}(\mathbf{s}')$ ▷ $\mathbf{s} = P \cdot \mathbf{e}$
3: $\mathbf{e} \leftarrow P^{-1} \cdot \mathbf{s}$
4: **return** $\Phi_{n,t}^{-1}(\mathbf{e})$

Ohne Kenntnis von $D_{\mathcal{C}}$ ist die Entschlüsselung allgemein schwer möglich und ist auf das NP-vollständige Syndrom Decoding-Problem (Definition 2.4.2) reduzierbar. Die Sicherheit von McEliece und Niederreiter ist bei gleicher Parameterwahl äquivalent [18, 25], siehe Abschnitt 4.1.

3.3 CFS-Signaturschema

Eng verwandt mit asymmetrischer Verschlüsselung sind Verfahren, um digitale Signaturen zu erzeugen und zu überprüfen (siehe Abbildung 2). Die erzeugte Signatur kann unabhängig vom signierten Dokument veröffentlicht werden und dient dazu, die Urheberschaft des signierten Dokumentes zu bezeugen: Ausschließlich der Besitzer des privaten Schlüssels sollte in der Lage sein, eine gültige Signatur anzufertigen.

Eine Signatur setzt also eine Berechnung voraus, die der Signierende mit vertretbarem Aufwand durchführen kann, andere jedoch nicht, und die dennoch von allen Beteiligten auf ihre Richtigkeit überprüft werden kann. Courtois, Finiasz und Sendrier schlugen im Jahr 2001 vor, als solche Berechnung das effiziente Decodieren von Syndromen zu Fehlervektoren zu verwenden [10].

Das von diesen Autoren vorgestellte Verfahren basiert auf Niederreiter (Abschnitt 3.2): Ein CFS-Schlüsselpaar $(K^{\text{pub}}, K^{\text{priv}})$ ist ein Niederreiter-Schlüsselpaar mit den bekannten Parametern $m, t, n \in \mathbb{N}$ mit der Einschränkung $n = 2^m$. Wie bisher ist k die Dimension des geheimen linearen Codes $\mathcal{C}$.

Bei der Konstruktion des Signaturschemas gehen wir von der Existenz einer kryptographisch sicheren Hashfunktion $h\colon \mathbb{F}_2^{\ell} \to \mathbb{F}_2^{n-k}$ für beliebige $\ell \in \mathbb{N}_0$ aus. Anstatt

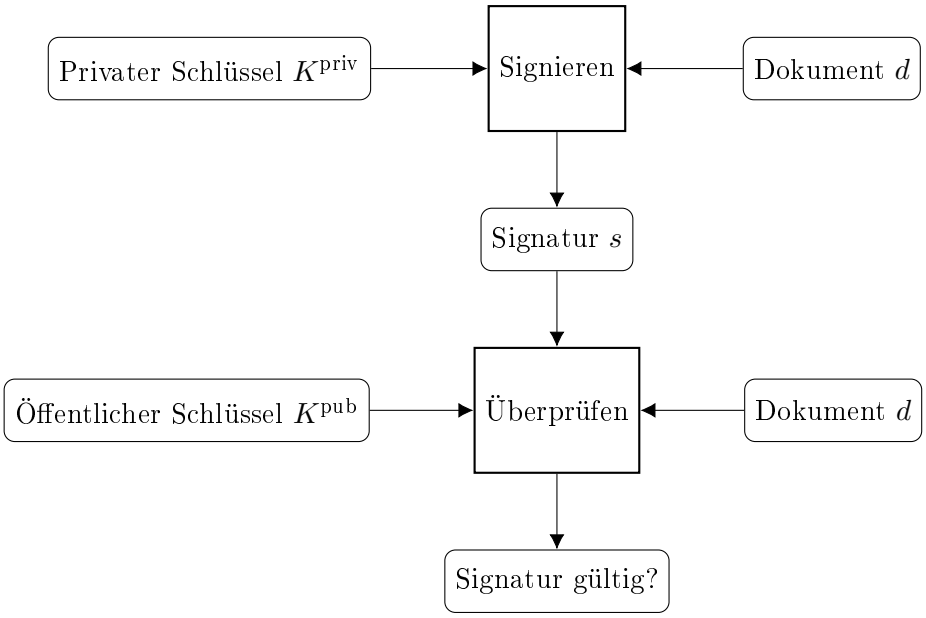

Abbildung 2: Struktur eines kryptographischen Signaturschemas

eine Signatur für ein Dokument $\mathbf{d} \in \mathbb{F}_2^\ell$ zu erzeugen, werden wir eine Signatur für $h(\mathbf{d})$ bestimmen. Die Grundidee besteht darin, $h(\mathbf{d})$ als Syndrom zu interpretieren, über Niederreiter zu entschlüsseln und den resultierende Klartext $\mathbf{m}$ als Signatur zu verwenden. Zum Verifizieren muss lediglich die Niederreiter-Verschlüsselung angewandt werden. Da allerdings von den insgesamt $2^{n-k} = 2^{mt} = n^t$ möglichen Syndromen nur $\binom{n}{t} \approx \frac{n^t}{t!}$ Syndrome zu Fehlervektoren mit Gewicht t gehören, ist es unwahrscheinlich ($p \approx \frac{1}{t!}$), dass ein gegebener Wert $h(\mathbf{d})$ entschlüsselt werden kann [10, S. 162]. Die von Courtois, Finiasz und Sendrier vorgeschlagene Lösung besteht darin, eine Zahl i an $\mathbf{z} = h(\mathbf{d})$ anzuhängen, bis eine der erzeugten Kombinationen $h(\mathbf{z} \,\|\, i)$ ein entschlüsselbares Syndrom darstellt:

Algoritmus 3.3.1 CFS — Signieren

Eingabe: Dokument $\mathbf{d} \in \mathbb{F}_2^\ell$
Ausgabe: Signatur $(\mathbf{m}, i) \in \left(\mathbb{F}_2^{n-k} \times \mathbb{N}\right)$

1: $\mathbf{z} \leftarrow h(\mathbf{d})$
2: $i \leftarrow 0$
3: **loop**
4: $\quad \mathbf{s} \leftarrow h(\mathbf{z} \,\|\, i)$
5: $\quad$ **if** $\mathbf{s}$ ist entschlüsselbar **then**
6: $\quad\quad \mathbf{m} \leftarrow \textsc{NiederreiterDecrypt}(K^{\text{priv}}, \mathbf{s})$ ▷ Algorithmus 3.2.3
7: $\quad\quad$ **return** $(\mathbf{m}, i)$
8: $\quad i \leftarrow i + 1$

Ohne Kenntnis von K^{priv} ist das Erzeugen einer gültigen Signatur hinreichend schwer. Zum Überprüfen einer Signatur $(\mathbf{m}, i)$ muss lediglich überprüft werden, ob $h\big(h(\mathbf{d}) \,\|\, i\big)$ dem Syndrom von $\Phi_{n,t}(\mathbf{m})$ entspricht:

Algoritmus 3.3.2 CFS — Verifizieren

Eingabe: Dokument $\mathbf{d} \in \mathbb{F}_2^{\ell}$, Signatur $(\mathbf{m}, i) \in \big(\mathbb{F}_2^{n-k} \times \mathbb{N}\big)$
Ausgabe: „Signatur gültig“ oder „Signatur ungültig“

1: $\mathbf{z} \leftarrow h(\mathbf{d})$
2: $\mathbf{s} \leftarrow \textsc{NiederreiterEncrypt}(K^{\text{pub}}, \mathbf{m})$ ▷ Algorithmus 3.2.2
3: **if** $\mathbf{s} = h\big(\mathbf{z} \,\|\, i\big)$ **then**
4: **return** Signatur gültig.
5: **else**
6: **return** Signatur ungültig.

Das Schema kann auch ohne die Abbildung $\Phi_{n,t}$ verwendet werden, was längere Signaturen und eine kürzere Laufzeit zur Folge hat. Die Signatur ist dann $(\mathbf{e}, i)$, wobei $\mathbf{e}$ der in Algorithmus 3.2.3 berechnete Fehlervektor und nicht der Klartext $\mathbf{m}$ ist.

3.4 Syndrombasiertes Zero-Knowledge-Protokoll

Jacques Stern stellte 1993 einen Zero-Knowledge-Beweis vor, der auf dem Syndrome Decoding-Problem (siehe Definition 2.4.2) basiert [29]. Ein solches Protokoll dient dazu, einen sogenannten Verifizierer (*verifier*) davon zu überzeugen, dass der Beweiser (*prover*) über ein bestimmtes Wissen verfügt und — unter der Annahme, dass dieses Wissen geheim ist — somit auch über eine bestimmte Identität. Aus diesem Grund wird das Verfahren auch als *Stern's identification scheme* bezeichnet. Wichtig ist auch, dass der Verifizierer im Rahmen des Protokolls keine Kenntnisse erlangt, die es ihm ermöglichen würden, sich gegenüber Dritten als der Beweiser auszugeben.

Das Protokoll funktioniert iterativ: In jedem Schritt werden zwei Nachrichten vom Beweiser an den Verifizierer übertragen und eine Nachricht in die entgegengesetzte Richtung. In jedem Schritt hat ein Beweiser, der über das geheime Wissen nicht verfügt, eine Chance von $\frac{2}{3}$, dem Verifizierer diese Tatsache zu verheimlichen; nach r Schritten besteht also eine Wahrscheinlichkeit von $\left(\frac{2}{3}\right)^r$, dass der Verifizierer fälschlicherweise davon ausgeht, dass der Beweiser die Identität besitzt, als die er

sich ausgibt.

Der private Schlüssel, also das geheime Wissen, über das der Beweiser verfügt, ist dabei ein Fehlervektor $\mathbf{s} \in \mathbb{F}_2^n$ mit $\mathrm{wt}(\mathbf{s}) = p$, wobei $n, p, k \in \mathbb{N}$ Parameter des Protokolls sind. Außerdem ist allen Parteien eine zufällig erzeugte Kontrollmatrix $H \in \mathbb{F}_2^{(n-k)\times n}$ für einen (n, k)-linearen Code $\mathcal{C}$ (siehe Definition 2.2.4) bekannt. Der öffentliche Schlüssel $\mathbf{i} \in \mathbb{F}_2^{n-k}$, die Identität des Beweisers, ist das Syndrom $\mathbf{i} = H \cdot \mathbf{s}$ des privaten Schlüssels $\mathbf{s}$. Den privaten Schlüssel $\mathbf{s}$ aus $\mathbf{i}$ zu rekonstruieren ist somit eine Instanz des Syndrome Decoding-Problems.

Anmerkung. *Nicht jede Instanz des Syndrome Decoding-Problems ist schwer. Dennoch gehen wir davon aus, dass eine zufällig konstruierte Instanz mit geeigneten Parametern im Durschnitt schwer ist [2].*

Das Protokoll macht außerdem Gebrauch von sogenannten Commitments:

Definition 3.4.1. *Ein* Commitment *für ein mathematisches Objekt (beispielsweise ein Tupel) ist ein kryptographischer Hashwert von einer Binärdarstellung des Objekts.*

Durch ein Commitment für ein Object o kann ein Kommunikationsteilnehmer zeigen, dass er o bereits zum Zeitpunkt des Commitments kannte, auch wenn er o selbst erst später veröffentlicht.

Wir unterscheiden beim Beweiser zwischen dem sogenannten *honest prover*, der den privaten Schlüssel $\mathbf{s}$ kennt, und einem *cheating prover*, welcher nicht über die geheimen Informationen verfügt, aber sich gegenüber dem Verifizierer dennoch als Beweiser ausgibt. Allgemein ist die Berechnung von $\mathbf{c}_3$ für einen *cheating prover* nicht möglich, was er aber unter Umständen verheimlichen kann: Es lassen sich verschiedene Strategien definieren, um mit der oben genannten Wahrscheinlichkeit von $\left(\frac{2}{3}\right)^r$ nach r Runden ohne Kenntnis von $\mathbf{s}$ einen Verifizierer vom Gegenteil zu überzeugen, welche beispielsweise von Overbeck und Sendrier zusammengefasst wurden [25, S. 103].

Das Protokoll ist in Algorithmus 3.4.1 dargestellt und wird als Zero-Knowledge bezeichnet, da ein Angreifer keine Informationen über $\mathbf{s}$ erhält, die es ihm ermöglichen würden, sich als Beweiser auszugeben: Wie bereits erläutert wurde, kann $\mathbf{s}$ aus

i nur schwer rekonstruiert werden. Das Protokoll selbst ermöglicht ebenfalls keine Rückschlüsse auf **s**. Die Sicherheit ist daher mit der der Verschlüsselungsgleichung von Niederreiter vergleichbar.

Algoritmus 3.4.1 Zero-Knowledge-Protokoll von Stern

Beweiser	**Verifizierer**
Wähle zufällig einen Vektor $\mathbf{y} \in \mathbb{F}_2^n$ und eine Permutation $\sigma \in S_n$.	
$\mathbf{c}_1 \leftarrow (\sigma, H \cdot \mathbf{y})$	
$\mathbf{c}_2 \leftarrow (\sigma(\mathbf{y}))$	
$\mathbf{c}_3 \leftarrow (\sigma(\mathbf{y} \oplus \mathbf{s}))$	
Übertrage Commitments für $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$.	
	Sende zufällig $b \in \{0, 1, 2\}$.
if $b = 0$ **then**	
Übertrage $\mathbf{y}, \sigma$.	
if $b = 1$ **then**	
Übertrage $\mathbf{y} \oplus \mathbf{s}, \sigma$.	
if $b = 2$ **then**	
Übertrage $\sigma(\mathbf{y}), \sigma(\mathbf{s})$.	
	if $b = 0$ **then**
	Prüfe $\mathbf{c}_1, \mathbf{c}_2$.
	if $b = 1$ **then**
	Prüfe $\mathbf{c}_1, \mathbf{c}_3$. [1]
	if $b = 2$ **then**
	Prüfe $\mathbf{c}_2$, $\mathbf{c}_3$ und $\mathrm{wt}(\sigma(\mathbf{s})) = p$.

3.5 Syndrombasierte Hashfunktion

Sei $\mathcal{C}$ ein (n, k)-linearer Code über $\mathbb{F}_2$ für $n, k \in \mathbb{N}$. Dann existieren 2^{n-k} unterschiedliche Syndrome und $\binom{n}{w}$ Fehlervektoren $\mathbf{v} \in \mathbb{F}_2^n$ mit $\mathrm{wt}(\mathbf{v}) = w$ für $w \in \mathbb{N}$. Wählt man n, k und w so, dass $n - k < \mathcal{L}_{n,w}$ gilt (siehe Abschnitt 3.2), so gilt auch

$$2^{n-k} < \binom{n}{w},$$

[1] Der Verifizierer kann $H \cdot \mathbf{y}$ berechnen, ohne $\mathbf{y}$ zu kennen: $H \cdot \mathbf{y} = (H \cdot (\mathbf{y} \oplus \mathbf{s})) \oplus \mathbf{i}$

also existieren mehr Fehlervektoren mit Gewicht w als Syndrome von $\mathcal{C}$. Ist es unter dieser Voraussetzung immer noch schwer, von Syndromen auf die jeweiligen Fehlervektoren zu schließen, so kann aus einem solchen Code eine kryptographisch sichere Hashfunktion konstruiert werden [2, 25]. Im Folgenden sei $H \in \mathbb{F}_2^{(n-k)\times n}$ eine Kontrollmatrix von $\mathcal{C}$ unter der Annahme, dass $\mathcal{C}$ zufällig erzeugt wurde und kein effizienter Algorithmus zur Syndromdecodierung von $\mathcal{C}$ bekannt ist. Wir definieren die Kompressionsfunktion $f\colon \mathbb{F}_2^{\mathcal{L}_{n,w}} \to \mathbb{F}_2^{n-k}$ mit

$$f(\mathbf{x}) = H \cdot \Phi_{n,w}(\mathbf{x}).$$

Gemäß der weit verbreiteten Merkle-Damgård-Konstruktion [12] leiten wir daraus eine kryptographische Hashfunktion $h\colon \mathbb{F}_2^{\ell} \to \mathbb{F}_2^{n-k}$ für $\mathbf{m} \in \mathbb{F}_2^{\ell}$ für beliebige $\ell \in \mathbb{N}_0$ ab:

Algoritmus 3.5.1 Syndrombasierte Hashfunktion

Eingabe: Nachricht $\mathbf{m} \in \mathbb{F}_2^{\ell}$
Ausgabe: Hashwert $\mathbf{h} \in \mathbb{F}_2^{n-k}$

1: **if** ℓ ist kein Vielfaches von $\mathcal{L}_{n,w}$ **then**
2: Füge Padding zu $\mathbf{m}$ hinzu, sodass ℓ ein Vielfaches von $\mathcal{L}_{n,w}$ ist.
3: Teile $\mathbf{m}$ auf in Blöcke $\mathbf{x}_1, \ldots, \mathbf{x}_{\ell/\mathcal{L}_{n,w}}$ der Länge $\mathcal{L}_{n,w}$.
4: Initialisiere $\mathbf{h}_0$ mit einer Konstanten.
5: **for** i **from** 1 **to** $\ell/\mathcal{L}_{n,w}$ **do**
6: $\mathbf{h}_i \leftarrow f\big(\mathbf{h}_{i-1} \,\|\, \mathbf{x}_i\big)$
7: **return** $\mathbf{h}_{\ell/\mathcal{L}_{n,w}}$

Augot, Finiasz und Sendrier stellen neben dieser Funktion zudem eine modifizierte Variante *Fast Syndrome Based* (FSB) vor, auf die im Rahmen dieser Arbeit allerdings nicht weiter eingegangen werden soll. Zudem bewiesen sie die Sicherheit dieses Verfahrens formal [2].

3.6 Kryptographisch sicherer Pseudozufallszahlengenerator

Seien $\mathcal{C}, n, k, w$ und H definiert wie in Abschnitt 3.5. Wählt man n, k und w so, dass

$$2^{n-k} > \binom{n}{w}$$

gilt, so existieren mehr Syndrome von $\mathcal{C}$ als Fehlervektoren mit Gewicht w. Insbesondere gilt auch $n - k > \mathcal{L}_{n,w}$ (siehe Abschnitt 3.2). Ist es unter dieser Voraussetzung immer noch schwer, von Syndromen auf die jeweiligen Fehlervektoren zu schließen, so kann aus einem solchen Code ein Generator für kryptographisch sichere Pseudozufallszahlen konstruiert werden [14, 25]. Wir definieren die Expansionsfunktion $f\colon \mathbb{F}_2^{\mathcal{L}_{n,w}} \to \mathbb{F}_2^{n-k}$ mit

$$f(\mathbf{x}) = H \cdot \Phi_{n,w}(\mathbf{x}).$$

Ausgehend von f erzeugen wir Pseudozufallszahlen $\mathbf{r}_i \in \mathbb{F}_2^{n-k-\mathcal{L}_{n,w}}$ für $i \in \{1, \ldots, \ell\}$, indem wir $f(\mathbf{x}_i)$ für Eingabewerte $\mathbf{x}_i \in \mathbb{F}_2^{\mathcal{L}_{n,w}}$ berechnen und das Ergebnis, einen Vektor der Länge $n - k$, in den neuen Eingabewert $\mathbf{x}_{i+1}$ mit Länge $\mathcal{L}_{n,w}$ und die Ausgabe $\mathbf{r}_i$ mit Länge $n - k - \mathcal{L}_{n,w}$ aufteilen. Der erste Eingabewert $\mathbf{x}_0$, der sogenannte Seed, muss selbst eine kryptographisch sichere Zufallszahl sein. Jeder Schritt dieses Algorithmus erzeugt also eine Pseudozufallszahl $\mathbf{r}_i$ der Länge $n - k - \mathcal{L}_{n,w}$.

Algoritmus 3.6.1 Syndrombasierter Pseudozufallszahlengenerator

Eingabe: $\mathbf{x}_i \in \mathbb{F}_2^{\mathcal{L}_{n,w}}$
Ausgabe: Zufallszahl $\mathbf{r}_i \in \mathbb{F}_2^{n-k-\mathcal{L}_{n,w}}$, nächste Eingabe $\mathbf{x}_{i+1} \in \mathbb{F}_2^{\mathcal{L}_{n,w}}$

1: $(s_1, \ldots, s_{n-k}) \leftarrow f(\mathbf{x}_i)$
2: $\mathbf{x}_{i+1} \leftarrow (s_1, \ldots, s_{\mathcal{L}_{n,w}})$
3: $\mathbf{r}_i \leftarrow (s_{\mathcal{L}_{n,w}+1}, \ldots, s_{n-k})$
4: **return** $(\mathbf{r}_i, \mathbf{x}_{i+1})$

Fischer und Stern zeigten 1996, dass dieses Verfahren zu brechen genauso schwierig ist, wie das Syndrome Decoding-Problem für entsprechende Parameter zu lösen, daher gelten die erzeugten Pseudozufallszahlen als kryptographisch sicher [14].

4 Sicherheit

Seit das McEliece-Verfahren 1978 vorgestellt wurde, wurden zahlreiche Angriffe auf dieses und ähnliche codebasierte Systeme vorgestellt. Eine aktuelle und ausführliche Auflistung entscheidender Angriffe auf McEliece beziehungsweise Niederreiter wurde von Bernstein et al. zusammengestellt [7]. Dieses Kapitel soll einen Überblick über bekannte Angriffe und sicherheitsrelevante Aspekte geben und erklären, weshalb codebasierte Kryptographie trotz einiger Schwächen nach aktuellem Kenntnisstand sicher ist, und welche Voraussetzungen notwendig sind, um besagte Schwächen zu vermeiden.

4.1 Äquivalenz von McEliece und Niederreiter

Wie bereits Li et al. gezeigt haben, ist die Sicherheit von McEliece und Niederreiter äquivalent [18]. Der Beweis soll an dieser Stelle auf ähnliche Weise dargestellt werden und erfolgt analog zur Reduktion des General Decoding-Problems (Definition 2.4.1) auf das Syndrome Decoding-Problem (Definition 2.4.2) und umgekehrt.

Satz 4.1.1. *Die Sicherheit von McEliece und Niedereiter ist bei gleicher Parameter- und Codewahl äquivalent: Ist ein Angreifer in der Lage, eines dieser Systeme zu brechen, so ist es ihm ebenfalls möglich, das jeweils andere System zu brechen.*

Beweis. Wir nehmen zunächst an, dass ein Angreifer Niederreiter brechen kann, also aus einem gegebenen öffentlichen Niederreiter-Schlüssel (H^{pub}, t) und Geheimtext

$$\mathbf{c}_{\text{N}} = H^{\text{pub}} \cdot \Phi_{n,t}(\mathbf{m}_{\text{N}})$$

den Klartext $\mathbf{m}_{\text{N}}$ rekonstruieren kann. Basierend auf dieser Annahme konstruieren wir einen Angriff auf McEliece. Gegeben seien der öffentliche McEliece-Schlüssel (G^{pub}, t) und der Geheimtext

$$\mathbf{c}_{\text{ME}} = \mathbf{m}_{\text{ME}} \cdot G^{\text{pub}} + \mathbf{e},$$

gesucht ist der Klartext $\mathbf{m}_{\text{ME}}$.

1. Berechne H^{pub} mit $G^{\text{pub}} \cdot \left(H^{\text{pub}}\right)^\intercal = 0$.
2. Berechne $\mathbf{c}_{\text{N}} = H^{\text{pub}} \cdot \mathbf{c}_{\text{ME}}$. Dann gilt $\mathbf{e} = \Phi_{n,t}(\mathbf{m}_{\text{N}})$.
3. Berechne $\mathbf{m}_{\text{N}}$ durch einen Angriff auf Niederreiter mit dem Schlüssel (H^{pub}, t) und dem Geheimtext $\mathbf{c}_{\text{N}}$.
4. Löse das lineare Gleichungssystem $G^\intercal \cdot \mathbf{m}_{\text{ME}} = \mathbf{c}_{\text{ME}} - \Phi_{n,t}(\mathbf{m}_{\text{N}})$.

McEliece ist also höchstens so sicher wie Niederreiter für gleiche Parameter. Nehmen wir umgekehrt an, dass ein Angreifer die Verschlüsselungsgleichung von McEliece brechen kann. Wir konstruieren einen Angriff auf Niederreiter mit einem öffentlichen Schlüssel (H^{pub}, t) und dem Geheimtext $\mathbf{c}_{\text{N}}$ folgendermaßen:

1. Berechne G^{pub} mit $G^{\text{pub}} \cdot \left(H^{\text{pub}}\right)^\intercal = 0$.
2. Finde einen beliebigen Vektor $\mathbf{c}_{\text{ME}}$ mit $H^{\text{pub}} \cdot \mathbf{c}_{\text{ME}} = \mathbf{c}_{\text{N}}$.
3. Aus $H^{\text{pub}} \cdot \mathbf{c}_{\text{ME}} = H^{\text{pub}} \cdot \Phi_{n,t}(\mathbf{m}_{\text{N}})$ folgt $H^{\text{pub}} \cdot (\mathbf{c}_{\text{ME}} - \Phi_{n,t}(\mathbf{m}_{\text{N}})) = 0$, also gilt $\mathbf{c}_{\text{ME}} - \Phi_{n,t}(\mathbf{m}_{\text{N}}) \in \mathcal{C}$. Dann existiert ein $\mathbf{m}_{\text{ME}}$ mit $\mathbf{m}_{\text{ME}} \cdot G^{\text{pub}} = \mathbf{c}_{\text{ME}} - \Phi_{n,t}(\mathbf{m}_{\text{N}})$, was äquivalent zu $\mathbf{c}_{\text{ME}} = \mathbf{m}_{\text{ME}} \cdot G^{\text{pub}} + \Phi_{n,t}(\mathbf{m}_{\text{N}})$ ist.
4. Berechne $\mathbf{m}_{\text{ME}}$ durch einen Angriff auf McEliece mit dem Schlüssel (G^{pub}, t) und dem Geheimtext $\mathbf{c}_{\text{ME}}$.
5. Setze $\mathbf{m}_{\text{N}} = \Phi_{n,t}^{-1}(\mathbf{c}_{\text{ME}} - \mathbf{m}_{\text{ME}} \cdot G)$.

Niederreiter ist also auch nicht sicherer als McEliece. □

4.2 Parameterwahl für McEliece

McEliece schlug ursprünglich die Parameter $n = 1024, k = 524, t = 50$ vor [21], was zu einem öffentlichen Schlüssel mit einer Größe von 65 Kilobyte — oder, wenn eine systematische Matrix verwendet wird (Abschnitt 4.6), 32 Kilobyte — führt. McEliece schätzte die Sicherheit, also den Aufwand, um die Verschlüsselung eines gegebenen Geheimtextes mit diesen Parametern zu brechen, auf 2^{65} Operationen. Neuere Angriffe haben den Aufwand auf etwa 2^{60} reduziert [22, S. 4].

Das *National Institute of Standards and Technology* (NIST) schätzt die Sicherheit von RSA mit 2048-Bit-Schlüsseln auf 2^{112}, was von derselben Institution bis 2030 als hinreichend betrachtet wird [3]. Niebuhr et al. schlagen $n = 2440, k = 1877, t = 50$

vor, um dieselbe Sicherheit zu erreichen, was einer Größe des öffentlichen Schlüssels von 560 Kilobyte beziehungsweise 129 Kilobyte bei der Verwendung einer systematischen Matrix entspricht [22]. Um der Sicherheit von AES-256 zu entsprechen, also mindestens 2^{256} Operationen zu erfordern, schlugen Bernstein et al. öffentliche Schlüssel mit Größen von mehr als 1000 Kilobyte vor [7].

Der private Schlüssel ist im Fall von McEliece wesentlich kleiner als der öffentliche, da sich der verwendete Goppa-Code sowie die geheime Permutationsmatrix wesentlich effizienter darstellen lassen als eine Matrix unbekannter Struktur.

4.3 Angriffe auf den Schlüssel

Prinzipiell lassen sich Angriffe auf asymmetrische Verschlüsselungsverfahren wie McEliece und Niederreiter in Angriffe auf den geheimen Schlüssel und Angriffe auf den Geheimtext aufteilen [24, S. 294]. Dieser Abschnitt bietet einen Überblick über bekannte Angriffe auf den Schlüssel im Fall von Niederreiter und McEliece, also Methoden, um den privaten Schlüssel zu rekonstruieren.

4.3.1 Niederreiter mit GRS-Codes

Niederreiter schlug ursprünglich vor, sogenannte *Generalized Reed-Solomon-Codes* in seinem Kryptosystem zu verwenden [23]. Wenige Jahre später zeigten Sidelnikov und Shestakov, dass die Verwendung von diesen GRS-Codes unsicher ist [28]. Sie zeigten, dass es möglich ist, einen alternativen privaten Schlüssel in Polynomialzeit zu finden, der die Entschlüsselung ermöglicht. Dieser Angriff lässt sich jedoch nicht auf Goppa-Codes und somit nicht auf McEliece beziehungsweise Niederreiter mit Goppa-Codes anwenden [11, S. 15ff.].

4.3.2 Schwache Goppa-Codes

Loidreau und Sendrier zeigten, dass sich öffentliche Schlüssel, die auf einem Goppa-Code mit binärem Generatorpolynom beruhen, sowohl erkennen als auch brechen lassen. Binäre Generatorpolynome sind jene, bei denen sämtliche Koeffizienten aus $\mathbb{F}_2$ stammen [11, 20]. Solche Polynome sollten im Rahmen der Schlüsselerzeugung

folglich vermieden werden, um diesen Angriff zu verhindern.

4.4 Angriffe auf den Geheimtext

Die einfachste Möglichkeit, den Klartext im Fall von McEliece aus dem Geheimtext ohne Kenntnis des privaten Schlüssels zu ermitteln, besteht darin, alle möglichen Eingaben mit dem verwendeten öffentlichen Schlüssel zu verschlüsseln. Dieser Angriff ist offensichtlich wenig vielversprechend, da bis zu 2^k Nachrichten verschlüsselt werden müssen, bevor der gesuchte Klartext gefunden wird. An dieser Stelle sollen daher effizientere Angriffe vorgestellt werden.

4.4.1 Generalized Information-Set-Decoding

Angriffe durch sogenanntes *Information-Set-Decoding* (ISD) wurden bereits von McEliece vorgeschlagen [21]. Diese Methode benötigt exponentiell viel Zeit, ist aber dennoch bedeutend schneller als ein Brute-Force-Angriff.

Die Idee besteht darin, zufällig k Spalten $\mathcal{I} \subset \{1, \ldots, n\}$ der $k \times n$-Generatormatrix G des öffentlichen McEliece-Schlüssels zu wählen und zu hoffen, dass diese Positionen des Geheimtextes nicht fehlerbehaftet sind. Wir bezeichnen mit $G_\mathcal{I}$ die Matrix, die sich aus den Spalten $\mathcal{I}$ von G ergibt, und mit $\mathbf{c}_\mathcal{I}$ beziehungsweise $\mathbf{e}_\mathcal{I}$ entsprechend die Werte des Geheimtextes $\mathbf{c}$ beziehungsweise Fehlervektors $\mathbf{e}$. Ist $G_\mathcal{I}$ invertierbar, so gilt

$$\mathbf{c}_\mathcal{I} = \mathbf{m} \cdot G_\mathcal{I} + \mathbf{e}_\mathcal{I}.$$

Sind die gewählten Stellen tatsächlich nicht fehlerbehaftet, so gilt $\mathbf{e}_\mathcal{I} = 0$ und damit

$$\mathbf{m} = \mathbf{c}_\mathcal{I} \cdot G_\mathcal{I}^{-1}. \tag{4.1}$$

Die Wahrscheinlichkeit, dass die gewählten Positionen $\mathcal{I}$ fehlerfrei sind, beträgt aber nur circa

$$\left(1 - \frac{t}{n}\right)^k,$$

McEliece schätzt daher die Anzahl der benötigten Operationen auf

$$k^3 \cdot \left(1 - \frac{t}{n}\right)^{-k}.$$

Um herauszufinden, ob die gewählten Stellen tatsächlich fehlerfrei sind, also ob $\mathbf{e}_\mathcal{I} = 0$ gilt, kann überprüft werden, ob $\mathrm{wt}(\mathbf{c} - \mathbf{m} \cdot G) = t$ gilt.

Wenige Jahre später schlugen Lee und Brickell eine Verallgemeinerung dieses Angriffs vor [17], die als *Generalized Information-Set-Decoding* (GISD) bezeichnet wird: Sie ersetzten die Bedingung $\mathbf{e}_\mathcal{I} = 0$ durch $\mathrm{wt}(\mathbf{e}_\mathcal{I}) \leq j$ für ein kleines $j \in \mathbb{N}$, wodurch die Wahrscheinlichkeit steigt, entsprechende Positionen $\mathcal{I}$ zufällig zu finden: Von den k gewählten Positionen dürfen nun bis zu j Stellen fehlerbehaftet sein. Offensichtlich gilt nun Gleichung 4.1 nicht mehr, wir können jedoch aus $\mathbf{e}_\mathcal{I}$ immer noch $\mathbf{m}$ rekonstruieren:

$$\mathbf{m} = (\mathbf{c}_\mathcal{I} - \mathbf{e}_\mathcal{I}) \cdot G_\mathcal{I}^{-1} \tag{4.2}$$

Ebenso wie $\mathcal{I}$ kann auch $\mathbf{e}_\mathcal{I}$ erraten werden, es existieren genau $\sum_{i=0}^{j} \binom{i}{k}$ solche Fehlervektoren. Um herauszufinden, ob ein gewählter Fehlervektor $\mathbf{e}_\mathcal{I}$ mit $\mathrm{wt}(\mathbf{e}_\mathcal{I}) \leq j$ der gesuchte Vektor ist, muss überprüft werden, ob der resultierende (gesamte) Fehlervektor $\mathbf{e}$ ein Gewicht von t hat. Für $\mathbf{e}$ gilt

$$\mathbf{e} = \mathbf{c} - \mathbf{m} \cdot G = \mathbf{c} - \left((\mathbf{c}_\mathcal{I} - \mathbf{e}_\mathcal{I}) \cdot G_\mathcal{I}^{-1}\right) \cdot G$$

Falls ein Vektor $\mathbf{e}_\mathcal{I}$ mit $\mathrm{wt}(\mathbf{e}_\mathcal{I}) \leq j$ existiert, so dass $\mathrm{wt}(\mathbf{e}) = t$ gilt, so folgt aus Gleichung 4.2 der gesuchte Klartext $\mathbf{m}$.

Algoritmus 4.4.1 Generalized Information-Set-Decoding

Eingabe: Geheimtext $\mathbf{c} \in \mathbb{F}_2^n$, öffentlicher Schlüssel (G, t), Parameter $j \in \mathbb{N}$
Ausgabe: Klartext $\mathbf{m} \in \mathbb{F}_2^k$

```
1: loop
2:     Wähle I ⊂ {1, ..., n} mit |I| = k so, dass G_I invertierbar ist.
3:     for i from 0 to j do
4:         for each e_I ∈ F_2^k mit wt(e_I) = i do
5:             m ← (c_I − e_I) · G_I^{-1}
6:             if wt(c − m · G) = t then
7:                 return m
```

4.4.2 Finding-Low-Weight-Codeword-Angriff

Der sogenannte *Finding-Low-Weight-Codeword*-Angriff kann McEliece unter der Voraussetzung brechen, dass ein Angreifer in der Lage ist, für einen gegebenen linearen Code das Codewort mit geringstem Hamming-Gewicht zu bestimmen [16, S. 22].

Sei $G \in \mathbb{F}_q^{k \times n}$ eine Generatormatrix eines (n, k)-linearen Codes $\mathcal{C}$ mit minimaler Distanz d, $\mathbf{e} \in \mathbb{F}_q^n$ ein Fehlervektor mit $1 \leq \mathrm{wt}(\mathbf{e}) < \frac{d}{2}$ und $\mathbf{m} \in \mathbb{F}_q^k$ eine Nachricht. Setze $\mathbf{c} = \mathbf{m} \cdot G + \mathbf{e}$ wie bei McEliece üblich.

Wir konstruieren einen neuen $(n, k+1)$-linearen Code $\mathcal{C}'$ anhand der Generatormatrix

$$G' = \begin{pmatrix} G \\ \mathbf{c} \end{pmatrix}.$$

Anmerkung. *Offensichtlich ist* $\mathbf{c}$ *linear unabhängig zu* $\mathcal{C}$*, da* $\mathbf{c}$ *ansonsten ein Codewort wäre, was wegen* $1 \leq \mathrm{wt}(\mathbf{e}) < \frac{d}{2}$ *unmöglich ist. Die Dimension von* $\mathcal{C}'$ *ist also tatsächlich* $k + 1$*.*

Konstruktionsbedingt sind sowohl $\mathbf{m} \cdot G$ als auch $\mathbf{c}$ Codewörter von $\mathcal{C}'$, folglich muss auch die Differenz $\mathbf{c} - \mathbf{m} \cdot G = \mathbf{e}$ ein Codewort sein. Da das Gewicht von $\mathbf{e}$ kleiner als $\frac{d}{2}$ ist, ist $\mathbf{e}$ sogar das Codewort mit dem geringsten Gewicht (abgesehen von 0). Ist ein Angreifer in der Lage, für einen gegebenen linearen Code $\mathcal{C}'$ ein solches Codewort $\mathbf{e}$ mit minimalen Gewicht zu finden, so kann er folglich auch McEliece brechen.

Es ist kein Algorithmus bekannt, der dieses Problem in Polynomialzeit löst, dennoch ist diese Methode effizienter als ein Brute-Force-Angriff [11, S. 25ff.]. Kobara vermutet, dass der Rechenaufwand mindestens $\binom{n}{k+1} / \binom{n-t}{k+1}$ beträgt [16, S. 22].

4.4.3 Angriff durch teilweise bekannten Klartext

Kennt ein Angreifer einen Teil des Klartextes, so erleichtert dies Angriffe auf den verbleibenden, unbekannten Klartext erheblich. An dieser Stelle soll eine Verallgemeinerung des Beispiels von Kobara und Imai dargestellt werden [16, S. 22f.].

Gesucht ist $\mathbf{m} = (m_1, \ldots, m_k)$ für eine gegebene Generatormatrix $G \in \mathbb{F}_2^{k \times n}$ und ein $\mathbf{c} = \mathbf{m} \cdot G + \mathbf{e}$. Seien $\mathcal{I} \subseteq \{1, \ldots, k\}$ die Indizes, für die die Werte m_i bekannt sind. Wir konstruieren einen Vektor $\mathbf{m}' = (m'_1, \ldots, m'_k)$, indem wir $m'_i = m_i$ für $i \in \mathcal{I}$ und $m'_i = 0$ für $i \notin \mathcal{I}$ setzen.

Wir bezeichnen mit G_u die Matrix, die sich aus G ergibt, wenn die Zeilen an den Indizes $\mathcal{I}$ entfernt werden. Mit $\mathbf{m}_u$ bezeichnen wir den Vektor der unbekannten Werte. Dann gilt:

$$\begin{aligned} \mathbf{c} &= \mathbf{m} \cdot G + \mathbf{e} \\ \mathbf{c} &= \mathbf{m}' \cdot G + \mathbf{m}_u \cdot G_u + \mathbf{e} \\ \mathbf{c}_u \coloneqq \mathbf{c} - \mathbf{m}' \cdot G &= \mathbf{m}_u \cdot G_u + \mathbf{e} \end{aligned}$$

Da wir $\mathbf{c}_u = \mathbf{c} - \mathbf{m}' \cdot G$ trivial berechnen können, verbleibt lediglich das Problem $\mathbf{c}_u = \mathbf{m}_u \cdot G_u + \mathbf{e}$. Die Dimension des linearen Codes, der durch G_u aufgespannt wird, ist jetzt nur noch $k - |\mathcal{I}|$. Ist diese Dimension eine kleine Konstante, so ist die Berechnung der unbekannten Werte $\mathbf{m}_u$ in Polynomialzeit möglich [16, S. 23].

4.4.4 Verfälschung des Klartextes

Auch ohne den eigentlichen Klartext $\mathbf{m}$ zu kennen, der mit McEliece gemäß $\mathbf{c} = \mathbf{m} \cdot G + \mathbf{e}$ verschlüsselt wurde, kann ein Angreifer einen Geheimtext $\mathbf{c}'$ erzeugen, der durch Entschlüsselung zu $\mathbf{m}' = \mathbf{m} + \delta\mathbf{m}$ für ein beliebig gewähltes $\delta\mathbf{m}$ führt [16, S. 24]:

$$\mathbf{c}' = \mathbf{m}' \cdot G + \mathbf{e} = (\mathbf{m} + \delta\mathbf{m}) \cdot G + \mathbf{e} = \mathbf{c} + \delta\mathbf{m} \cdot G$$

Dies funktioniert im Fall von Niederreiter durch die Codierung von Nachrichten in Fehlervektoren mit konstantem Gewicht nicht [11, S. 38].

4.4.5 Related-Message-Angriff

Gegeben seien zwei Geheimtexte $\mathbf{c}_1 = \mathbf{m}_1 \cdot G + \mathbf{e}_1$, $\mathbf{c}_2 = \mathbf{m}_2 \cdot G + \mathbf{e}_2$, die durch die Verschlüsselung von Klartexten $\mathbf{m}_1, \mathbf{m}_2$ nach McEliece entstanden sind. Kennt ein Angreifer den Wert $\delta\mathbf{m} = \mathbf{m}_1 + \mathbf{m}_2$, was wegen der binären Darstellung $\mathbf{m}_1 \oplus \mathbf{m}_2$ entspricht, so führt *Generalized Information-Set-Decoding* (siehe Abschnitt 4.4.1) mit hoher Wahrscheinlichkeit schnell zu den gesuchten Klartexten. Dies ist insbesondere dann der Fall, wenn der Angreifer weiß, dass $\mathbf{m}_1 = \mathbf{m}_2$ gilt, also dass dieselbe

Nachricht mehrmals, aber mit verschiedenen Fehlervektoren, übertragen wurde. Mit Kenntnis von $\delta\mathbf{m}$ kann nun auch $\delta\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ berechnet werden:

$$\begin{aligned}\delta\mathbf{e} &= \mathbf{e}_1 + \mathbf{e}_2 \\ &= (\mathbf{c}_1 - \mathbf{m}_1 \cdot G) + (\mathbf{c}_2 - \mathbf{m}_2 \cdot G) \\ &= \mathbf{c}_1 + \mathbf{c}_2 - (\mathbf{m}_1 + \mathbf{m}_2) \cdot G \\ &= \mathbf{c}_1 + \mathbf{c}_2 - \delta\mathbf{m} \cdot G\end{aligned}$$

Der Angreifer wendet nun GISD auf $\mathbf{c}_1$ oder $\mathbf{c}_2$ an und wählt als Indizes $\mathcal{I}$ zufällig k Stellen, an denen $\delta\mathbf{e}$ den Wert null hat. An diesen Positionen haben mit hoher Wahrscheinlichkeit sowohl $\mathbf{e}_1$ als auch $\mathbf{e}_2$ den Wert null, da das Gewicht t der Fehlervektoren sehr viel kleiner als n ist. Somit führt GISD schnell zu dem jeweiligen Klartext $\mathbf{m}_1$ beziehungsweise $\mathbf{m}_2$ [16, S. 23].

4.5 IND-CPA- und IND-CCA-Sicherheit

Ein wichtiges Kriterium eines asymmetrischen Verschlüsselungsverfahrens, das oft als *semantische Sicherheit* bezeichnet wird, ist sogenannte IND-CPA-Sicherheit.

Definition 4.5.1. IND-CPA-Sicherheit *(Indistinguishability under chosen-plaintext attack) bedeutet, dass jeder Angreifer,*

1. *dem ein öffentlicher Schlüssel K^{pub} vorliegt,*
2. *der zwei beliebige Klartexte $\mathbf{m}_1$ und $\mathbf{m}_2$ produziert, und*
3. *der für einen gegebenen Geheimtext $\mathbf{c}$ entscheiden kann, ob dieser durch Verschlüsselung von $\mathbf{m}_1$ oder durch Verschlüsselung von $\mathbf{m}_2$ erzeugt wurde,*

entweder für die Erzeugung von $\mathbf{m}_1$ und $\mathbf{m}_2$ (Schritt 2) oder für die Entscheidung, zu welchem Klartext $\mathbf{c}$ gehört (Schritt 3), exponentiell viel Zeit benötigt, gemessen an einem Sicherheitsparameter des öffentlichen Schlüssels K^{pub}.

Ein IND-CPA-sicheres System gibt also nur vernachlässigbare Informationen über den Klartext preis. Offensichtlich trifft dies auf McEliece in der ursprünglichen Form nicht zu, da ein Angreifer $\mathbf{c} - \mathbf{m}_1 \cdot G$ und $\mathbf{c} - \mathbf{m}_2 \cdot G$ berechnen kann

und über das Gewicht des resultierenden Vektors die verwendete Nachricht ermitteln kann [24, S. 293].

Wie Nojima et al. zeigten, genügt es, vor der Verschlüsselung eine zufällige Bitfolge an den Klartext anzuhängen, um IND-CPA-Sicherheit zu erreichen: Anstatt Nachrichten der Länge k zu verschlüsseln, werden Nachrichten der Länge k' für ein $k' \in \mathbb{N}$ mit $k' < k$ zufällig zu Vektoren der Länge k ergänzt, welche anschließend mit McEliece verschlüsselt werden [24]. Offensichtlich verhindert dies beispielsweise den Related-Message-Angriff aus Abschnitt 4.4.5, da $\delta\mathbf{m}$ unbekannt ist.

Für ein stärkeres Sicherheitsverständnis benötigen wir außerdem eine Definition sogenannter Entschlüsselungsorakel:

Definition 4.5.2 (Entschlüsselungsorakel). *Im Kontext eines Angriffs auf einen Geheimtext* $\mathbf{c}$ *ist ein* Entschlüsselungsorakel *ein Algorithmus, der vom Angreifer beliebig gewählte Geheimtexte (abgesehen von* $\mathbf{c}$*) entschlüsselt.*

Basierend auf dem Konzept eines Entschlüsselungsorakels lässt sich ein stärkerer Sicherheitsbegriff als IND-CPA definieren:

Definition 4.5.3. IND-CCA-Sicherheit *(Indistinguishability under adaptive chosen ciphertext attack) entspricht IND-CPA-Sicherheit (Definition 4.5.1), mit dem Unterschied, dass der Angreifer in Schritt 2 und 3 Zugriff auf ein Entschlüsselungsorakel hat.*

Da McEliece nicht IND-CPA-sicher ist, kann es auch nicht IND-CCA-sicher sein. Ein Angriff unter Zuhilfenahme eines Entschlüsselungsorakels lässt sich beispielsweise aus Abschnitt 4.4.4 ableiten: Da beide Klartexte vom Angreifer in Schritt 2 frei gewählt werden können, sind $\mathbf{m}_1$, $\mathbf{m}_2$ und insbesondere auch $\delta\mathbf{m} = \mathbf{m}_2 - \mathbf{m}_1$ bekannt, folglich kann der Angreifer aus dem ihm präsentierten Geheimtext $\mathbf{c}$ in Schritt 3 einen anderen Geheimtext $\mathbf{c}' = \mathbf{c} + \delta\mathbf{m} \cdot G^{\text{pub}}$ berechnen. Mithilfe des Entschlüsselungsorakels entschlüsselt er $\mathbf{c}'$ und erhält $\mathbf{m}'$, also entweder $\mathbf{m}_1$, wenn $\mathbf{c}$ durch $\mathbf{m}_2$ entstanden ist, oder $\mathbf{m}_2$, wenn $\mathbf{c}$ durch $\mathbf{m}_1$ entstanden ist. Die gesuchte Nachricht $\mathbf{m}$ ist dann $\mathbf{m} = \mathbf{m}' + \delta\mathbf{m}$ [16, S. 24]. Ein ähnlicher Angriff lässt sich auch für die soeben vorgestellte, IND-CPA-sichere Variante von McEliece konstruieren, die somit ebenfalls nicht IND-CCA-sicher ist.

Es existieren verschiedene Methoden, um — unter der Annahme, dass das McEliece-Problem (Definition 3.1.1) schwer zu lösen ist — basierend auf McEliece ein IND-CCA-sicheres Schema zu konstruieren, welche beispielsweise von Kobara und Imai ausführlich diskutiert wurden. Dieselben Autoren stellten zudem drei neue Möglichkeiten dar, die sie als α-, β- und γ-Konvertierungen für McEliece bezeichnen, welche IND-CCA-Sicherheit erreichen und wesentlich effizienter als andere bekannte Konvertierungen sind [16]. Diese Konvertierungen verhindern beziehungsweise erschweren die in den Abschnitten 4.4.3, 4.4.4 und 4.4.5 beschriebenen Angriffe, deren praktische Relevanz dadurch verschwindet.

4.6 Reduktion der Schlüsselgröße

Das Niederreiter-Kryptosystem (Abschnitt 3.2) wird häufig mit einer systematischen Matrix als öffentlichem Schlüssel verwendet.

Definition 4.6.1. *Eine* systematische $a \times b$-Matrix *ist eine Matrix, deren erste a Spalten (falls $a \leq b$) beziehungsweise erste b Zeilen (falls $a > b$) eine Einheitsmatrix bilden.*

Dazu muss lediglich die Matrix M während der Schlüsselerzeugung (Algorithmus 3.2.1) entsprechend gewählt werden, beispielsweise durch das Gaußsche Eliminationsverfahren. Der Vorteil dieser Variante liegt darin, dass die ersten k Spalten der $(n-k) \times n$-Kontrollmatrix nicht übertragen werden müssen, da sie die Einheitsmatrix bilden, was die Größe des öffentlichen Schlüssels von $(n-k) \cdot n$ auf $(n-k) \cdot k$ Bits reduziert. Dies hat nach bisherigen Erkenntnissen keinen negativen Einfluss auf die Sicherheit des Verfahrens [7, 25].

Im Fall von McEliece ist die Verwendung einer systematischen Generatormatrix schwieriger, da das Produkt $\mathbf{m} \cdot G$ für eine Nachricht $\mathbf{m} \in \mathbb{F}_2^k$ und eine systematische Generatormatrix $G \in \mathbb{F}_2^{k \times n}$, deren erste k Spalten die Einheitsmatrix bilden, offensichtlich Informationen über den Klartext verrät, selbst wenn anschließend ein Fehlervektor addiert wird. Dennoch erlauben IND-CCA-sichere Konvertierungen für McEliece die Verwendung systematischer Generatormatrizen, was die Schlüsselgröße auf $k \cdot (n-k)$ Bits reduziert [25, S. 128f.].

4.7 Angriffe auf das CFS-Signaturschema

Für Signaturschemata ist das relevanteste Kriterium, an dem die Sicherheit des Verfahrens gemessen wird, der Aufwand, den ein Angreifer benötigt, um ohne Kenntnis des geheimen Schlüssels eine gültige Signatur zu erzeugen. Für die ursprünglich von Courtois, Finiasz und Sendirer vorgeschlagenen Parameter $n = 2^{16}, t = 9$ für das CFS-Signaturschema (Abschnitt 3.3) gaben die Autoren eine Sicherheit von etwa 2^{83} Operationen an. Wie später gezeigt wurde, ist es bei diesen Parametern jedoch möglich, bereits nach etwa 2^{59} Operationen eine gültige Signatur zu finden [25, S. 114]. Es ist kein Angriff in Polynomialzeit bekannt, sodass sich diese Schwäche durch eine entsprechende Erhöhung der Parameter ausgleichen lässt.

4.8 Einfluss von Quantencomputern

Ein verbreitetes Hilfsmittel zur Konstruktion von Quantenalgorithmen ist der Grover-Algorithmus, der mit hoher Wahrscheinlichkeit eine Nullstelle einer Funktion $f\colon \mathbb{F}_2^b \to \mathbb{F}_2$ durch etwa $\sqrt{2^b}$ Auswertungen von f findet. Diese Wahrscheinlichkeit kann durch mehrfache Ausführung weiter erhöht werden. So lässt sich beispielsweise ein naiver Angriff auf einen McEliece-Geheimtext $\mathbf{c}$ mit dem öffentlichen Schlüssel (G^{pub}, t) durch eine Funktion $f\colon \mathbb{F}_2^k \to \mathbb{F}_2$ mit

$$f(\mathbf{m}) = \begin{cases} 0, & \text{falls } \operatorname{wt}(\mathbf{c} - \mathbf{m} \cdot G^{\text{pub}}) = t \\ 1, & \text{sonst} \end{cases}$$

konstruieren, deren Nullstelle offensichtlich der gesuchte Klartext ist. Der Grover-Algorithmus reduziert den Aufwand gegenüber demselben Angriff ohne Zuhilfenahme eines Quantencomputers von 2^k auf etwa $\sqrt{2^k} = 2^{k/2}$.

Es ist zudem möglich, den Grover-Algorithmus auf Funktionen mit mehr als einer Nullstelle zu verallgemeinern: Wenn f genau r Nullstellen besitzt, dann findet die allgemeine Form des Grover-Algorithmus mit hoher Wahrscheinlichkeit eine Nullstelle in nur $\sqrt{2^b/r}$ Iterationen [9]. Wie Bernstein zeigt, erlaubt es diese Variante des Algorithmus, den Grover-Algorithmus auch auf Information-Set-Decoding (Abschnitt 4.4.1) anzuwenden [5, S. 77f.]. Sei $S = \mathcal{P}_k(\{1, \ldots, n\})$ die Menge aller k-

elementigen Teilmengen von $\{1, \ldots, n\}$. Wir definieren die Funktion $f_{\mathrm{ISD}} \colon S \to \mathbb{F}_2$:

$$f_{\mathrm{ISD}}(\mathcal{I}) = \begin{cases} 0, & \text{falls } G_{\mathcal{I}} \text{ invertierbar ist und } \operatorname{wt}\left(\mathbf{c} - \left(\mathbf{c}_{\mathcal{I}} \cdot G_{\mathcal{I}}^{-1}\right) \cdot G\right) = t \\ 1, & \text{sonst.} \end{cases}$$

Wie oben bereits erklärt wurde, führt Information-Set-Decoding zu einer Menge nicht fehlerbehafteter Stellen $\mathcal{I}$ von $\mathbf{c}$, aus denen der Klartext trivial rekonstruiert werden kann. Eine solche Menge $\mathcal{I}$ ist folglich eine Nullstelle von f_{ISD}, und diese kann mit dem verallgemeinerten Grover-Algorithmus gefunden werden. Dabei ist egal, welche Nullstelle $\mathcal{I}$ von f_{ISD} tatsächlich gefunden wird, da jede Nullstelle gleichermaßen zum Ergebnis führt.

Nach Bernsteins Berechnungen muss n etwa verdoppelt werden, um gegen Angriffe unter Zuhilfenahme von Quantencomputern dieselbe Sicherheit zu bieten wie zuvor gegen klassische Computer. Da k ungefähr gleichermaßen mitwächst, führt dies zu einer Vervierfachung der Größe des öffentlichen Schlüssels [5, S. 74].

Es ist kein nennenswert effizienterer Angriff als dieser bekannt, auch nicht unter Zuhilfenahme größerer Quantencomputer. Die Auswirkungen aller bekannten Angriffe können durch eine entsprechende Erhöhung der Schlüsselgröße kompensiert werden, was codebasierte Kryptographie zu einem vielversprechenden Kandidaten für Post-Quanten-Kryptographie macht [7].

5 Zusammenfassung

Codebasierte kryptographische Verfahren bilden eine vielversprechende Alternative zu etablierten asymmetrischen Schemata wie RSA. Als nach wie vor sicher gelten McEliece (Abschnitt 3.1) und die duale Variante von Niederreiter (Abschnitt 3.2) mit einer Vielzahl möglicher Konvertierungen zu IND-CCA-sicheren Systemen (Abschnitt 4.5). Trotz der Geschwindigkeit dieser Verfahren und ihrer vermuteten Resistenz gegen die Beschleunigung durch Quantenparallelität zukünftiger Computer stellen die im Vergleich zu aktuellen Methoden sehr großen Schlüsselpaare ein Hindernis dar. So legten Bernstein et al. 2017 der amerikanischen Behörde NIST eine IND-CCA-sichere Variante von McEliece beziehungsweise Niederreiter zusammen mit beispielhaften Parametern vor [7], welche in öffentlichen Schlüsseln mit einer Größe von etwa einem Megabyte und privaten Schlüsseln von über zehn Kilobyte resultieren – RSA-Schlüssel sind dagegen meist nur wenige Kilobyte groß.

Auch verwandte Verfahren wie beispielsweise syndrombasierte Hashfunktionen (Abschnitt 3.5), die Erzeugung kryptographisch sicherer Pseudozufallszahlen (Abschnitt 3.6) sowie das Zero-Knowledge-Protokoll von Stern (Abschnitt 3.4), welche auf Kontrollmatrizen linearer Codes basieren, weisen geringe Komplexität und nach aktuellen Kenntnissen dennoch eine hohe Sicherheit auf, welche sich auf NP-vollständige Probleme zurückführen lässt. Angriffe in Polynomialzeit sind daher kaum denkbar.

Selbst unter Zuhilfenahme von Quantencomputern sind im Gegensatz zu etablierten Verfahren wie RSA und Diffie-Hellman keine Angriffe bekannt, die die Sicherheit von den in Kapitel 3 vorgestellten codebasierten Verfahren maßgeblich beeinträchtigen würden, obwohl entsprechende Versuche unternommen wurden. Dennoch verbleibt die Möglichkeit, dass zukünftige Bemühungen effizientere Angriffe ermöglichen werden, welche zu weiteren Erhöhungen der Schlüsselgröße oder — schlimmstenfalls — zur Unbrauchbarkeit dieser Verfahren führen. Je näher die Forschung der vollständigen Erschließung von Quantencomputern kommt, desto mehr Aufmerksamkeit wird Post-Quanten-Kryptographie erfahren, und desto mehr werden wir über ihre Sicherheit, Effizienz und Brauchbarkeit erfahren.

A Endliche Körper

Endliche Körper werden auch als als Galoiskörper in Anerkennung des französischen Mathematikers Évariste Galois bezeichnet. Ein endlicher Körper $\mathbb{F}_q$ mit q Elementen existiert genau dann, wenn q eine Primzahlpotenz ist, daher werden endliche Körper häufig als $\mathbb{F}_{p^n}$ oder $\mathrm{GF}(p^n)$ für eine Primzahl p und eine positive natürliche Zahl $n \in \mathbb{N}_+$ notiert.

Arithmetik in $\mathbb{F}_p$

Ist q eine Primzahl p, gilt also in obiger Notation $q = p$ und $n = 1$, so ist die Arithmetik in $\mathbb{F}_p$ besonders einfach zu beschreiben. Addition und Multiplikation können in $\mathbb{Z}/p\mathbb{Z}$ mittels bekannter modularer Arithmetik durchgeführt werden.

Besonders relevant ist der Körper $\mathbb{F}_2$, der ausschließlich die jedem Körper innewohnenden Elemente 0 und 1 enthält, und in dem folgende Aussagen gelten:

$$\begin{aligned} 0+0&=0 & 0\cdot 0&=0\\ 0+1&=1 & 0\cdot 1&=0\\ 1+1&=0 & 1\cdot 1&=1 \end{aligned}$$

Alle anderen endlichen Körper enthalten Elemente, die von den neutralen Elementen 0 und 1 verschieden sind. In $\mathbb{F}_3$ gelten beispielsweise folgende Regeln:

$$\begin{aligned} 0+0&=0 & 1+1&=2 & 0\cdot 0&=0 & 1\cdot 1&=1\\ 0+1&=1 & 1+2&=0 & 0\cdot 1&=0 & 1\cdot 2&=2\\ 0+2&=2 & 2+2&=1 & 0\cdot 2&=0 & 2\cdot 2&=1 \end{aligned}$$

Arithmetik in $\mathbb{F}_{p^n}$

Ist $q = p^n$ eine Primzahlpotenz mit $n > 1$, so gibt es bis auf Isomorphie genau einen Körper $\mathbb{F}_{p^n}$. Eine Repräsentation des Körpers wird durch ein normiertes, irreduzibles Polynom $f \in \mathbb{F}_p[x]$ mit Grad n festgelegt. Den durch f eindeutig festgelegten Körper notieren wir als $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(f)$.

Um die Arithmetik in einem solchen Körper zu vereinfachen, werden Zahlen aus

$\mathbb{F}_{p^n}$ als Polynome über $\mathbb{F}_p$ dargestellt. Dazu wird eine Zahl $a \in \mathbb{F}_{p^n}$ in ihre Ziffern $a_0, \ldots, a_{n-1} \in \mathbb{F}_p$ zerlegt und als Polynom $a(x) = a_0 + a_1 \cdot x + \cdots + a_{n-1} \cdot x^{n-1} \in \mathbb{F}_p[x]$ interpretiert. Die Addition zweier Zahlen kann nun analog zu gewöhnlicher Polynomaddition über $\mathbb{F}_p$ durchgeführt werden.

Für die Multiplikation zweier Zahlen $a, b \in \mathbb{F}_{p^n}$ werden ebenfalls die bekannten Regeln für Polynommultiplikation modulo f angewandt:

$$a \cdot b \coloneqq (a(x) \cdot b(x)) \mod f(x)$$

Die Reduktion anhand von $f(x)$ ist notwendig, da die Polynommultiplikation $a(x) \cdot b(x)$ Polynome höheren Grades zur Folge hätte. Da f vom Grad n ist, hat das resultierende Polynom maximal Grad $n-1$ und kann somit als Zahl in $\mathbb{F}_{p^n}$ dargestellt werden.

Beispielhaft betrachten wir den endlichen Körper $\mathbb{F}_2[x]/(x^3 + x + 1)$ mit $2^3 = 8$ Elementen. Wir stellen alle Elemente als Polynome über $\mathbb{F}_2$ dar:

Element aus $\mathbb{F}_{2^4}$	Darstellung in $\mathbb{F}_2[x]$	Darstellung als Ziffern aus $\mathbb{F}_2$
0	0	$(0,0,0)$
x^0	1	$(1,0,0)$
x^1	x	$(0,1,0)$
x^2	x^2	$(0,0,1)$
x^3	$1 + x$	$(1,1,0)$
x^4	$x + x^2$	$(0,1,1)$
x^5	$1 + x + x^2$	$(1,1,1)$
x^6	$1 + x^2$	$(1,0,1)$

Offensichtlich gilt gemäß Polynomaddition beispielsweise

$$x^0 + x^5 = (1+1) + x + x^2 = x + x^2 = x^4$$

und da $x^2 \cdot x^1 \equiv 1 + x \pmod{f(x)}$ gilt, gilt wie erwartet $x^1 \cdot x^2 = x^3$.

B Erweiterter Euklidischer Algorithmus

Seien a, b zwei Elemente eines euklidischen Rings. Der erweiterte euklidische Algorithmus (EEA) liefert den größten gemeinsamen Teiler beider Elemente sowie zwei Elemente u, v so, dass

$$\gcd(a, b) = u \cdot a + v \cdot b$$

gilt. Dieser Algorithmus kann verwendet werden, um das multiplikativ Inverse eines Elements a in einem endlichen Körper zu finden: Die Kongruenz $a \cdot u \equiv 1 \pmod{m}$ ist äquivalent zu $a \cdot u + m \cdot v = 1 = \gcd(a, m)$, also liefert der Algorithmus das multiplikativ Inverse u von a. Ist $\gcd(a, m) \neq 1$, so existiert kein multiplikativ Inverses von a modulo m.

Algoritmus B.1 Erweiterter Euklidischer Algorithmus für ganze Zahlen

Eingabe: $a, b \in \mathbb{N}$
Ausgabe: (x, u, v) mit $\gcd(a, b) = x = u \cdot a + v \cdot b$

1: $(s, s_{\text{alt}}) \leftarrow (0, 1)$
2: $(t, t_{\text{alt}}) \leftarrow (1, 0)$
3: $(r, r_{\text{alt}}) \leftarrow (b, a)$
4: **while** $r \neq 0$ **do**
5: Berechne q, r_{neu} mit $q \cdot r + r_{\text{neu}} = r_{\text{alt}}$ und $r_{\text{neu}} < r$.
6: $(r, r_{\text{alt}}) \leftarrow (r_{\text{neu}}, r)$
7: $(s, s_{\text{alt}}) \leftarrow (s_{\text{alt}} - q \cdot s, s)$
8: $(t, t_{\text{alt}}) \leftarrow (t_{\text{alt}} - q \cdot t, t)$
9: **return** $(r_{\text{alt}}, s_{\text{alt}}, t_{\text{alt}})$

Dieser Algorithmus kann durch minimale Veränderungen auch für Polynome verwendet werden (siehe Algorithmus B.2). Das Ergebnis ist dann nicht mehr eindeutig, da Multiplikation des größten gemeinsamen Teilers mit einer Konstante erneut zu einem größten gemeinsamem Teiler führt. Um auf ein eindeutiges Ergebnis zu kommen, kann r_{alt} zum Schluss durch den führenden Koeffizienten geteilt werden, was ein normiertes Polynom zur Folge hat.

Algoritmus B.2 Erweiterter Euklidischer Algorithmus für Polynome

Eingabe: $a, b \in \mathbb{K}[x]$

Ausgabe: (x, u, v) mit $\gcd(a, b) = x = u \cdot a + v \cdot b$

1: $(s, s_{\text{alt}}) \leftarrow (0, 1)$
2: $(t, t_{\text{alt}}) \leftarrow (1, 0)$
3: $(r, r_{\text{alt}}) \leftarrow (b, a)$
4: **while** $r \neq 0$ **do**
5: Berechne q, r_{neu} mit $q \cdot r + r_{\text{neu}} = r_{\text{alt}}$ und $\deg(r_{\text{neu}}) < \deg(r)$.
6: $(r, r_{\text{alt}}) \leftarrow (r_{\text{neu}}, r)$
7: $(s, s_{\text{alt}}) \leftarrow (s_{\text{alt}} - q \cdot s, s)$
8: $(t, t_{\text{alt}}) \leftarrow (t_{\text{alt}} - q \cdot t, t)$
9: **return** $(r_{\text{alt}}, s_{\text{alt}}, t_{\text{alt}})$

Literatur

[1] Aaronson, S. : BQP and the polynomial hierarchy. In: Schulman, L. J. (Hrsg.): *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, ACM, 141–150

[2] Augot, D. ; Finiasz, M. ; Sendrier, N. : *A Fast Provably Secure Cryptographic Hash Function*. Cryptology ePrint Archive, Report 2003/230, 2003. – `https://eprint.iacr.org/2003/230`

[3] Barker, E. B.: *NIST Special Publication 800-57: Recommendation for Key Management, Part 1: General*. National Institute of Standards and Technology, 2016

[4] Bernstein, D. J.: Introduction to post-quantum cryptography. In: Bernstein, D. J. (Hrsg.) ; Buchmann, J. (Hrsg.) ; Dahmen, E. (Hrsg.): *Post-Quantum Cryptography*. Springer-Verlag, 2009, S. 1–14

[5] Bernstein, D. J.: Grover vs. McEliece. In: Sendrier, N. (Hrsg.): *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings* Bd. 6061, Springer (Lecture Notes in Computer Science), 73–80

[6] Bernstein, D. J.: List Decoding for Binary Goppa Codes. In: Chee, Y. M. (Hrsg.) ; Guo, Z. (Hrsg.) ; Ling, S. (Hrsg.) ; Shao, F. (Hrsg.) ; Tang, Y. (Hrsg.) ; Wang, H. (Hrsg.) ; Xing, C. (Hrsg.): *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings* Bd. 6639, Springer (Lecture Notes in Computer Science), 62–80

[7] Bernstein, D. J. ; Chou, T. ; Lange, T. ; Maurich, I. von ; Misoczki, R. ; Niederhagen, R. ; Persichetti, E. ; Peters, C. ; Schwabe, P. ; Sendrier, N. ; Szefer, J. ; Wang, W. : *Classic McEliece: conservative code-based cryptography*. NIST Submission. `https://classic.mceliece.org/nist/mceliece-20171129.pdf`. Version: 11 2017

[8] Bernstein, E. ; Vazirani, U. V.: Quantum Complexity Theory. In: *SIAM J. Comput.* 26 (1997), Nr. 5, 1411–1473. http://dx.doi.org/10.1137/S0097539796300921. – DOI 10.1137/S0097539796300921

[9] Boyer, M. ; Brassard, G. ; Høyer, P. ; Tapp, A. : Tight Bounds on Quantum Searching. In: *Fortschritte der Physik* 46, Nr. 4-5, S. 493–505. http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. – DOI 10.1002/(SICI)1521–3978(199806)46:4/5<493::AID–PROP493>3.0.CO;2–P

[10] Courtois, N. ; Finiasz, M. ; Sendrier, N. : How to Achieve a McEliece-Based Digital Signature Scheme. In: Boyd, C. (Hrsg.): *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings* Bd. 2248, Springer (Lecture Notes in Computer Science), 157–174

[11] D. Engelbert, R. O. ; Schmidt, A. : *A Summary of McEliece-Type Cryptosystems and their Security.* Cryptology ePrint Archive, Report 2006/162, 2006. – https://eprint.iacr.org/2006/162

[12] Damgård, I. : A Design Principle for Hash Functions. In: Brassard, G. (Hrsg.): *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings* Bd. 435, Springer (Lecture Notes in Computer Science), 416–427

[13] Dummit, D. S. ; Foote, R. M.: *Abstract Algebra, 3rd Edition.* Wiley, 2004

[14] Fischer, J. ; Stern, J. : An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. In: Maurer, U. M. (Hrsg.): *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding* Bd. 1070, Springer (Lecture Notes in Computer Science), 245–255

[15] Jochemsz, E. : *Goppa Codes & the McEliece Cryptosystem*, VU University Amsterdam, Diss., 2002

[16] Kobara, K. ; Imai, H. : Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC. In: Kim, K. (Hrsg.): *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings* Bd. 1992, Springer (Lecture Notes in Computer Science), 19–35

[17] Lee, P. J. ; Brickell, E. F.: An Observation on the Security of McEliece's Public-Key Cryptosystem. In: Günther, C. G. (Hrsg.): *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings* Bd. 330, Springer (Lecture Notes in Computer Science), 275–280

[18] Li, Y. ; Deng, R. H. ; Wang, X. : On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. In: *IEEE Trans. Information Theory* 40 (1994), Nr. 1, 271–273. http://dx.doi.org/10.1109/18.272496. – DOI 10.1109/18.272496

[19] Ling, S. ; Xing, C. : *Coding theory: A First Course.* Cambridge University Press, 2004

[20] Loidreau, P. ; Sendrier, N. : Weak keys in the McEliece public-key cryptosystem. In: *IEEE Trans. Information Theory* 47 (2001), Nr. 3, 1207–1211. http://dx.doi.org/10.1109/18.915687. – DOI 10.1109/18.915687

[21] McEliece, R. J.: A public-key cryptosystem based on algebraic coding theory. In: *Jet Propulsion Laboratory DSN Progress Report* 42-44 (1978), S. 114–116

[22] Niebuhr, R. ; Meziani, M. ; Bulygin, S. ; Buchmann, J. : *Selecting Parameters for Secure McEliece-based Cryptosystems.* Cryptology ePrint Archive, Report 2010/271, 2010. – https://eprint.iacr.org/2010/271

[23] Niederreiter, H. : Knapsack-type cryptosystems and algebraic coding theory. In: *Problems of Control and Information Theory* 15 (2001), Nr. 1, S. 159–166

[24] Nojima, R. ; Imai, H. ; Kobara, K. ; Morozov, K. : Semantic security for the McEliece cryptosystem without random oracles. In: *Des. Codes Cryptography* 49 (2008), Nr. 1-3, 289–305. http://dx.doi.org/10.1007/s10623-008-9175-9. – DOI 10.1007/s10623–008–9175–9

[25] Overbeck, R. ; Sendrier, N. : Code-based cryptography. In: Bernstein, D. J. (Hrsg.) ; Buchmann, J. (Hrsg.) ; Dahmen, E. (Hrsg.): *Post-Quantum Cryptography*. Springer-Verlag, 2009, S. 95–145

[26] Rieffel, E. G. ; Polak, W. : An introduction to quantum computing for non-physicists. In: *ACM Comput. Surv.* 32 (2000), Nr. 3, 300–335. http://dx.doi.org/10.1145/367701.367709. – DOI 10.1145/367701.367709

[27] Shor, P. W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, IEEE Computer Society, 124–134

[28] Sidelnikov, V. M. ; Shestakov, S. O.: On insecurity of cryptosystems based on generalized Reed-Solomon codes. In: *Discrete Mathematics and Applications* 2 (1992), Nr. 4, S. 439–444

[29] Stern, J. : A New Identification Scheme Based on Syndrome Decoding. In: Stinson, D. R. (Hrsg.): *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings* Bd. 773, Springer (Lecture Notes in Computer Science), 13–21