Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Theoretische Informatik

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fractional Recoding for the Shamir Method

Diplomarbeit
zur Erlangung des akademischen Grades
Diplom-Informatiker

vorgelegt von: Daniel Adolph

am: 6. März 2009

Prüfer: Prof. Dr. Johannes Buchmann

Betreuer: Dr. Erik Dahmen

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 26. März 2009 _____

In der Kryptographie werden mathematische Verfahren entwickelt, mit denen Nachrichten ver- und entschlüsselt werden können, mit dem Ziel diese Nachrichten vor Dritten geheim zu halten. Diese mathematischen Verfahren werden auch Kryptosysteme genannt. Die Anwendung von elliptischen Kurven (EC) für die Kryptosysteme wurde erstmals 1985 vorgeschlagen. Im Vergleich zu anderen Kryptosystemen benötigen EC-Kryptosysteme bei gleichem Sicherheitsniveau wesentlich kürzere Schlüssel und ermöglichen somit eine effizientere Ver- und Entschlüsselung. Diese Eigenschaften machen die kryptographischen Verfahren mit elliptischen Kurven insbesondere für rechenschwache und speicherbegrenzte Endgeräte, wie Smart Cards und PDAs, interessant.

Die Sicherheit der EC-Kryptosysteme beruht auf der Tatsache, dass es einfach ist, ein Skalar mit einem elliptischen Kurvenpunkt zu multiplizieren, es aber praktisch unmöglich ist, aus dem Ergebnis dieser skalaren Punktmultiplikation das ursprüngliche Skalar zu berechnen. Für die Verifikation von digitalen Signaturen mit elliptischen Kurven ist die zeitaufwändigste Operation die Berechnung der mehrfachen Punktmultiplikation $k \cdot P + l \cdot Q$. Für die Berechnung dieser mehrfachen Punktmultiplikation stehen mit der Shamir Methode und der Interleave Methode zwei effiziente Evaluationsalgorithmen zur Verfügung. Diese Algorithmen können durch die Anwendung einer vorzeichenbehafteten Darstellung zur Basis Zwei signifikant beschleunigt werden, hierzu müssen dann aber einige Punkte im Evaluationsalgorithmus vorberechnet werden. Die fractional-window Rekodierung erlaubt es, die Anzahl der Punkte, die in der Interleave Methode vorberechnet werden müssen, exakt festzulegen. Dadurch kann der vorhandene Speicherplatz im Endgerät optimal ausgenutzt werden.

In dieser Arbeit wird die Möglichkeit untersucht, das fractional-window Rekodierungsverfahren auch mit der Shamir Methode zu kombinieren. Das Ziel ist es, ein fractional-window Shamir Verfahren zu entwickeln, das anhand der benötigten ECADD-Operationen der Interleave Methode mit der fractional-window Rekodierung überlegen ist. Die Effizienz dieses neuen Verfahrens steht dabei im Hintergrund, vielmehr geht es darum zu zeigen, dass solch eine Kombination prinzipiell möglich ist und Grundlagen für weitere Untersuchungen zu schaffen.

Diese Arbeit ist folgendermaßen aufgebaut: In Kapitel 1 werden kryptographische Grundbegriffe und -konzepte näher erläutert. Die elliptischen Kurven, Kryptosysteme mit elliptischen Kurven und die Vorteile der EC-Kryptographie werden in Kapitel 2 vorgestellt. In Kapitel 3 werden die Darstellungen von ganzen Zahlen und deren Hamming-Gewicht behandelt. Effiziente Evaluationsalgorithmen für die Berechnung der mehrfachen Punktmultiplikation werden in Kapitel 4 näher untersucht. Außerdem wird gezeigt, dass

durch die Anwendung von Darstellungen mit geringem Hamming-Gewicht diese Algorithmen beschleunigt werden können. Kapitel 5 behandelt Rekodierungsmethoden, mit denen Darstellungen mit einem geringen Hamming-Gewicht erzeugt werden können. In Kapitel 6 wird dargestellt wie sich die Evaluationsalgorithmen mit den Rekodierungsmethoden beschleunigen lassen. Zudem wird ein Vergleich zwischen der Interleave Methode und der Shamir Methode durchgeführt. Das fractional-window Shamir Verfahren wird in Kapitel 7 vorgestellt und sowohl für acht, als auch vier vorzuberechnende Punkte angepasst. In Kapitel 8, dem letzten Kapitel, werden schließlich die Schlussfolgerungen aufgeführt.

Tabellenverzeichnis

1. Laufzeitenvergleich der Signaturoperationen von EC-DSA, DSA und RSA auf einem Pentium Pro mit 200 MHz 16
2. Vergleich der Ausführungszeiten der Signaturoperationen von EC-DSA mit $E(\mathbb{F}_{2^m})$ und RSA mit $(\mathbb{F}_p)^*$ auf einem Sharp Zaurus SL-5500G 17
3. Beispielwerte für die $\mathcal{AHD}$ der Fractional-window Rekodierungsmethode . 44
4. Beispiele für die $\mathcal{AJHD}_k$ (JSF) für k Ganzzahlen 58
5. Beispiele für die $\mathcal{AJHD}_k$ (ltrJSF) für k Ganzzahlen 63
6. Abfolge der Fensterbreiten bei der DOT-JSF$_3$-Entwicklung 66
7. Abfolge der Fensterbreiten bei der nochmaligen Verwendung einer rekodierten Spalte während der DOT-JSF$_3$-Entwicklung 68
8. Vergleich zwischen der Shamir- und der Interleave Methode anhand der durchschnittlichen ECADD-Operationen 75
9. Abfolge der Fensterbreiten bei der fractional-window Methode für die Shamir Methode 76
10. Resultierende AJHD für das fractional-window Verfahren für die Shamir Methode 77
11. Resultierende AJHD für die DOT-JSF$_3$-Entwicklung 80
12. Resultierende AJHD bei der fractional-window Methode für die Shamir Methode mit acht vorzuberechnenden Punkten 83
13. Resultierende AJHD mit zusätzlichen Nullspalten bei der fractional-window Methode für die Shamir Methode mit acht vorzuberechnenden Punkten 86
14. Resultierende AJHD bei der fractional-window Methode für die Shamir Methode mit vier vorzuberechnenden Punkten 89
15. Resultierende AJHD mit zusätzlichen Nullspalten bei der fractional-window Methode für die Shamir Methode mit vier vorzuberechnenden Punkten 90
16. Vergleich zwischen dem Fractional-window Shamir Verfahren und der Interleave Methode anhand der durchschnittlichen ECADD-Operationen . . 92
17. Ergebnisse des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten 99
18. Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten 103
19. Ergebnisse des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten und zusätzlichen Nullspalten 106
20. Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten und zusätzlichen Nullspalten 108
21. Ergebnisse des FW-Shamir Verfahrens mit vier vorzuberechnenden Punkten 110
22. Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit vier vorzuberechnenden Punkten und zusätzlichen Nullspalten 131

Abbildungsverzeichnis

1. Singuläre elliptische Kurven 6
2. Punktaddition von Punkten einer elliptischen Kurve 7
3. Sonderfälle bei der Punktaddition von Punkten einer elliptischen Kurve . 9
4. Beispiel einer elliptischen Kurve über $\mathbb{F}_p$ 10

List of Algorithms

1. EC-Schlüsselpaarerzeugung . . . 18
2. EC-Diffie-Hellman Schlüsselaustausch (EC-DH) . . . 18
3. EC-ElGamal Public-Key Verschlüsselung . . . 19
4. EC-DSA Signaturerzeugung . . . 21
5. EC-DSA Signaturverifikation . . . 21
6. Dezimalzahl in Binärzahl darstellen . . . 24
7. Binärer Exponentiationsalgorithmus . . . 28
8. Exponentiationsalgorithmus für $\mathcal{D}$-Darstellungen . . . 30
9. Interleave Methode für $\mathcal{D}$-Darstellungen . . . 32
10. Shamir Methode für $\mathcal{D}$-Darstellungen . . . 35
11. Fixed-size-sliding-window Recoding Methode . . . 39
12. Sliding-window Recoding Methode . . . 41
13. Fractional-window Recoding Methode . . . 43
14. wNAF Recoding Methode . . . 46
15. MOF Recoding Methode . . . 48
16. wMOF Recoding Methode . . . 50
17. Joint Sparse Form Recoding Methode . . . 53
18. left-to-right Joint Sparse Form Recoding Methode . . . 62

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Kryptographie . . . 1
- 1.2. Symmetrische und asymmetrische kryptographische Verfahren . . . 1
- 1.3. Digitale Signaturen . . . 2
- 1.4. Kryptographisch sichere Hashfunktionen . . . 3

2. Elliptische Kurven **5**
- 2.1. Kurvengleichung . . . 5
- 2.2. Addition von Kurvenpunkten einer elliptischen Kurve . . . 6
  - 2.2.1. Punktaddition . . . 7
  - 2.2.2. Punktverdopplung . . . 8
- 2.3. Elliptische Kurven über Primkörper . . . 9
- 2.4. Die skalare Punktmultiplikation . . . 11
  - 2.4.1. Lim-Lee-Kämmung . . . 12
- 2.5. Kryptographie mit elliptischen Kurven . . . 13
  - 2.5.1. Das Problem des diskreten Logarithmus für elliptische Kurven . . . 13
  - 2.5.2. Domain Parameter für Elliptische Kurven . . . 14
  - 2.5.3. Vorteile von EC-Kryptographie . . . 16
- 2.6. Kryptosysteme mit elliptischen Kurven . . . 17
  - 2.6.1. Schlüsselpaarerzeugung . . . 17
  - 2.6.2. EC-Diffie-Hellman Schlüsselaustausch . . . 18
  - 2.6.3. EC-ElGamal Public-Key Verschlüsselung . . . 19
  - 2.6.4. Elliptische Kurven Digitale Signaturen Algorithmus . . . 20

3. Darstellungen ganzer Zahlen **23**
- 3.1. Binärdarstellung . . . 23
- 3.2. Vorzeichenbehaftete Darstellungen zur Basis Zwei . . . 24
- 3.3. Hamming-Gewicht von Darstellungen . . . 25

4. Evaluationsalgorithmen für die mehrfache Punktmultiplikation **28**
- 4.1. Binärer Exponentiationsalgorithmus . . . 28
- 4.2. Interleave Methode . . . 31
- 4.3. Shamir Methode . . . 33
- 4.4. Vorberechnung von elliptischen Kurvenpunkten . . . 36

5. Rekodierung des Skalars **38**
- 5.1. Window Recoding . . . 38
  - 5.1.1. Fixed-size-sliding-window Recoding Methode . . . 38
  - 5.1.2. Sliding-window Recoding Methode . . . 40
  - 5.1.3. Fractional-window Recoding Methode . . . 41
- 5.2. wNAF Darstellung . . . 45
- 5.3. MOF Darstellung . . . 47

5.3.1. wMOF Darstellung 48
5.4. Joint Sparse Form 52
5.4.1. Joint Sparse Form für zwei Ganzzahlen 52
5.4.2. Verallgemeinerte Joint Sparse Form 55
5.4.3. Entwicklung der Joint Sparse Form von links nach rechts 59
5.4.4. Die angepasste Joint Sparse Form DOT-JSF$_3$ für zwei Ganzzahlen 64

6. Effiziente Berechnung einer zweifachen Punktmultiplikation 71
6.1. Beschleunigung der Interleave Methode 71
6.2. Beschleunigung der Shamir Methode 72
6.3. Interleave Methode und die Shamir Methode im Vergleich 74

7. Fractional Recoding für die Shamir Methode 76
7.1. Fractional Recoding für die Shamir Methode mit zehn Punkten 76
7.2. Fractional Recoding für die Shamir Methode mit acht Punkten 82
7.3. Fractional Recoding für die Shamir Methode mit vier Punkten 87

8. Schlussfolgerung 91

Literatur 94

A. Tabellen für die FW-Sh Methode mit acht vorzuberechnenden Punkten 99
A.1. Resultierende AJHD 103
A.2. Zusätzliche Nullspalten 106
A.3. Resultierende AJHD mit zusätzlichen Nullspalten 108

B. Tabellen für die FW-Sh Methode mit vier vorzuberechnenden Punkten 110
B.1. Zusätzliche Nullspalten 131

1. Einleitung

1.1. Kryptographie

In der Kryptographie werden mathematische Verfahren entwickelt und analysiert, welche Nachrichten ver- und entschlüsseln können um diese vor Dritten geheim zu halten. Diese mathematischen Verfahren nennt man auch kryptographische Verfahren oder einfach Kryptosysteme. Kryptosysteme erfüllen je nach Anwendungszweck eines oder mehrere der folgenden Schutzziele [Sch05]:

Vertraulichkeit: Nur berechtigte Personen dürfen an den Inhalt der Nachricht gelangen.

Integrität: Die Nachricht darf nicht unbemerkt und nicht von unberechtigten Personen verändert werden.

Authentizität: Der Absender oder Urheber einer Nachricht muss eindeutig identifizierbar sein.

Verbindlichkeit: Der Absender oder Urheber einer Nachricht muss sich auch gegenüber Dritten nachweisen lassen.

1.2. Symmetrische und asymmetrische kryptographische Verfahren

Alice möchte zum Beispiel an Bob eine vertrauliche Nachricht schicken, die nur Bob lesen kann. Sie verschlüsselt dazu die Nachricht m mit einer Verschlüsselungsfunktion Enc und einem Verschlüsselungsschlüssel e. Eine Verschlüsselungsfunktion überführt den Klartext m der Nachricht in einen unsinnig erscheinenden Chiffretext c, wobei

$$Enc\,(e,m) = c$$

gilt. Alice schickt den Chiffretext c an Bob, dieser erhält den Klartext der Nachricht mit Hilfe der Entschlüsselungsfunktion Dec und dem Entschlüsselungsschlüssel d, durch

$$Dec\,(d,c) = m.$$

Sind die Schlüssel zum Ver- und Entschlüsseln gleich, oder lässt sich der eine Schlüssel leicht aus dem anderen berechnen, spricht man von einem symmetrischen Verfahren. Da die Schlüssel entweder gleich oder leicht voneinander zu berechnen sind, müssen Alice und Bob bei einem symmetrischen Verfahren vor der eigentlichen Kommunikation einen Schlüssel auf einem sicheren Weg austauschen und geheimhalten. Wollen in einer Gruppe von n Benutzern jeweils zwei Benutzer mittels eines symmetrischen kryptographischen

Verfahrens vertraulich kommunizieren, müssen insgesamt $n\,(n-1)\,/2$ Schlüssel untereinander ausgetauscht werden. Aus diesem Grund sind symmetrische Verfahren für größere Benutzergruppen unpraktisch, zudem stellt der geheime Schlüsselaustausch zwischen den einzelnen Benutzern ein zusätzliches Problem dar.

Im Gegensatz dazu werden bei den asymmetrischen kryptographischen Verfahren zwei unterschiedliche Schlüssel benutzt, wobei zwischen einem öffentlichen und einem privaten Schlüssel unterschieden wird. Der private Schlüssel muss geheim gehalten werden und lässt sich nur durch einen immensen Rechenaufwand aus dem öffentlichen Schlüssel berechnen. Der Aufwand hierfür ist so groß gewählt, dass es praktisch unmöglich ist den privaten von dem öffentlichen Schlüssel abzuleiten. Da der öffentliche Schlüssel nicht mehr geheim gehalten werden muss, entfällt das Problem des geheimen Schlüsselaustauschs. Jeder Teilnehmer kann seinen öffentlichen Schlüssel den Kommunikationspartnern offen bekannt geben oder in einem öffentlichen Verzeichnis eintragen lassen. Aus diesem Grund heißen asymmetrische Verfahren auch Public-Key Verfahren. Damit Alice an Bob eine vertrauliche Nachricht schicken kann, bezieht sie den öffentlichen Verschlüsselungsschlüssel e direkt von Bob oder aus einer öffentlichen Datenbank. Mit diesem verschlüsselt sie ihre Nachricht, welche nur Bob mit seinem privaten Entschlüsselungsschlüssel d wieder entschlüsseln kann.

Mithilfe der Public-Key Verfahren lässt sich zwar das Schlüsselaustauschproblem lösen, dennoch sind sie im Allgemeinen um Einiges langsamer als Symmetrische Verfahren. Um den einfachen Schlüsselaustausch und die Schnelligkeit der beiden Verfahren nutzen zu können, wird in der Praxis eine Kombination aus symmetrisch und asymmetrisch verwendet, das sogenannte Hybrid-Verfahren. Bei einem Hybrid-Verfahren wird der öffentliche Schlüssel nicht zur Verschlüsselung der Nachricht genutzt, sondern zum sicheren Austausch eines symmetrischen Sitzungsschlüssels, mit dem die Nachricht verschlüsselt wird. Möchte Alice wiederum eine vertrauliche Nachricht an Bob schicken, erzeugt sie zuerst einen zufälligen Sitzungsschlüssel k, mit dem sie die Nachricht m verschlüsselt und somit den Chiffretext c erhält. Den Sitzungsschlüssel k verschlüsselt Alice mit Bobs öffentlichem Verschlüsselungsschlüssel e. Den chiffrierten Sitzungsschlüssel und den Chiffretext c sendet sie an Bob, welcher mit Hilfe seines privaten Entschlüsselungsschlüssels d den Sitzungsschlüssel k erhält und mit diesem den Chiffretext c entschlüsseln kann [Buc03, Sch05].

1.3. Digitale Signaturen

Die Schutzziele Integrität, Authentizität und Verbindlichkeit können für elektronische Dokumente mittels digitaler Signaturen sichergestellt werden. Digitale Signaturen können mit der Unterschrift unter einem Dokument verglichen werden, wobei die Unterschrift für ein elektronisches Dokument Signatur genannt wird. Signaturverfahren nutzen öffentliche und private Schlüssel und bestehen aus einer Signaturerzeugungsfunktion

$Sign$ und einer Verifikationsfunktion $Verify$ für Signaturen. Die Signatur s für ein Dokument m erhält man mit der Funktion

$$Sign\,(sk,m) = s,$$

wobei sk der private Schlüssel des Unterzeichners ist. Ob wirklich der Besitzer des öffentlichen Schlüssels pk das Dokument m signiert hat, kann mit der Verifikationsfunktion

$$Verify\,(pk,s,m) \in \{\text{gültig, ungültig}\}$$

geprüft werden. Ist die Signatur s gültig, dann ist der Signierer der Nachricht auch authentisch. Nur wenn der private und öffentliche Schlüssel zusammen passen erhält man eine gültige Verifikation, und nur der Besitzer des öffentlichen Schlüssels kennt schließlich den geheimen privaten Schlüssel. Aus diesem Grund ist die Signatur auch verbindlich, da nur der Besitzer des privaten Schlüssels gültige Signaturen erzeugen kann. Ferner ist bei einer gültigen Signatur die Integrität des Dokumentes gewährleistet. Nur wenn das Dokument m bei der Signaturerzeugung und -verifikation exakt gleich ist, wird die Signatur als gültig erkannt und kann somit nicht unbemerkt verändert werden. Möchte nun Alice an Bob eine signierte Nachricht m schicken, berechnet sie mit der Signaturfunktion und ihrem privaten Schlüssel sk die Signatur s. Die Signatur zusammen mit der Nachricht schickt Alice an Bob, dieser kann mit dem öffentlichen Schlüssel pk von Alice und der Funktion $Verify$ prüfen, ob die Signatur gültig ist [Buc03, Sch05].

1.4. Kryptographisch sichere Hashfunktionen

Für digitale Signaturen und einige weitere Kryptosysteme werden kryptographisch sichere Hashfunktionen benötigt. Eine Hashfunktion ist eine Abbildung von beliebig langen Strings auf einen String mit fester Länge. Diese Abbildung ist nicht injektiv, daher kann es zu Kollisionen kommen, das heißt zwei unterschiedliche Strings werden auf den selben String abgebildet.

Definition. *Eine kryptographisch sichere Hashfunktion H ist eine nicht injektive Funktion*

$$H : \Sigma^* \to \Sigma^n, \quad n \in \mathbb{N},$$

die folgende Eigenschaften hat:

- *$H(x)$ lässt sich für alle $x \in \Sigma^*$ effizient berechnen. Es existiert aber kein effizientes Verfahren, dass für ein $s \in \Sigma^n$ ein $x \in \Sigma^*$ mit $H(x) = s$ finden kann.*

- *Es ist praktisch unmöglich für ein gegebenes $x \in \Sigma^*$ ein $x' \in \Sigma^*$ mit $x \neq x'$ zu finden, so dass es zu einer Kollision mit $H(x) = H(x')$ kommt.*

- *Es ist praktisch unmöglich eine Kollision $H(x) = H(x')$ für zwei beliebige Eingabestrings x,$x' \in \Sigma^*$ mit $x \neq x'$ zu finden.*

Mittels kryptographisch sicheren Hashfunktionen kann die Integrität von Dokumenten sichergestellt werden. Dazu bildet die Hashfunktion das Dokument auf einen eindeutigen String der Länge n ab, welcher auch der digitale Fingerabdruck des Dokuments genannt wird. In der Praxis wird n so groß gewählt, dass es statistisch gesehen unmöglich ist für zwei beliebige Dokumente den selben digitalen Fingerabdruck zu erhalten. Alice möchte an Bob eine Nachricht m schicken und will sicher gehen, dass eine Änderung der Nachricht bei der Übertragung erkannt wird. Dazu berechnet sie den Hashwert $H(m) = h$ und schickt diesen zusammen mit der Nachricht an Bob. Dieser berechnet für die empfangene Nachricht m' den Hashwert $H(m') = h'$, falls $h = h'$ gilt, ist die empfangene Nachricht aufgrund der Eigenschaften der Hashfunktion unverändert [Buc03, Eck07].

2. Elliptische Kurven

Ellipsen sind ebene Kurven im $\mathbb{R}^2$, die durch folgende quadratische Mittelpunktsgleichung definiert werden:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0 \quad \textit{mit } a,b,x,y \in \mathbb{R} \textit{ und } a,b \neq 0.$$

Durch den Versuch die Bogenlängen von Ellipsen zu berechnen entstanden die elliptischen Integrale. Die Umkehrungen dieser Integrale wurden als elliptische Funktionen bezeichnet. Aus der Theorie der elliptischen Funktionen entstanden die elliptischen Kurven und haben daher ihren Namen [Sil85]. Elliptische Kurven werden über einen Körper definiert und durch eine kubische Gleichung werden alle Punkte bestimmt, die auf dieser Kurve liegen. In der affinen Ebene haben die Punkte der Kurve eine x- und eine y-Koordinate, wobei die Koordinaten Elemente des Körpers sind. Falls die elliptische Kurve bestimmte Bedingungen erfüllt, kann man auf der Punktemenge dieser Kurve eine Verknüpfung definieren und mit dieser eine abelsche Gruppe bilden. Victor Miller [Mil86] und Neal Koblitz [Kob87b] schlugen unabhängig voneinander vor, diese Gruppenstruktur auch für Public-Key Kryptosysteme zu nutzen.

2.1. Kurvengleichung

Definition. *Eine elliptische Kurve E über einem Körper K ist definiert durch die Lösungsmenge der allgemeinen Weierstrassgleichung*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad \textit{mit } a_i \in K. \tag{1}$$

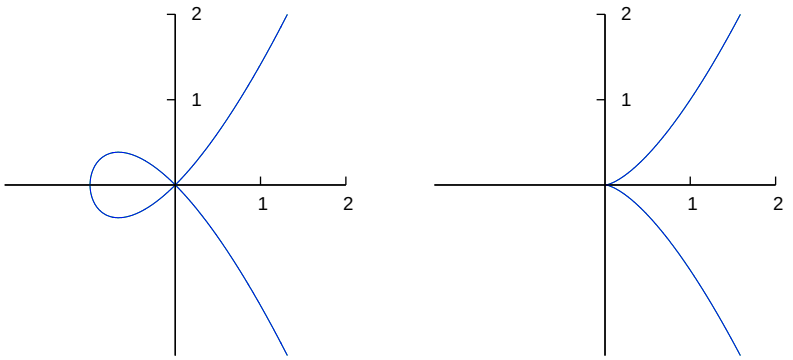
Die Menge aller Punkte $(x,y) \in K \times K$, die diese Gleichung erfüllen, vereinigt mit dem Punkt $\mathcal{O}$ im Unendlichen, wird mit $E(K)$ bezeichnet [Sil85].

Für die Anwendung in der Kryptographie wird zusätzlich gefordert, dass die elliptischen Kurven nicht singulär sein dürfen. Kurven die nicht singulär sind haben an jedem Punkt eine eindeutig definierte Tangente, solche Kurven werden auch als glatt bezeichnet.

Definition. *Eine elliptische Kurve über einem Körper K heißt singulär in dem Punkt (x,y) ∈ E(F), falls beide partiellen Ableitungen verschwinden. Dies ist der Fall, wenn in dem Punkt gleichzeitig folgende Gleichungen gelten [Kob98]:*

$$a_1y = 3x^2 + 2a_2x + a_4 \qquad\qquad 2y + a_1x + a_3 = 0$$

Beispiele für singuläre elliptische Kurven über dem Körper der reellen Zahlen sind in der Abbildung 1 zu sehen. Für den Newtonschen Knoten 1(a) mit der Gleichung $y^2 = x^3 + x^2$



(a) Newtonsche Knoten (b) Neilsche Parabel

Abbildung 1: Singuläre elliptische Kurven

gibt es im Punkt (0,0) keine eindeutige Tangente. Auch für die Gleichung $y^2 = x^3$ der Neilschen Parabel 1(b) existiert im Ursprung keine sinnvolle Tangente.

Ist die Charakteristik des Körpers K ungleich zwei und drei ($char(K) \neq 2, 3$), lässt sich die allgemeine Weierstrassgleichung in eine einfachere Darstellung umformen. Diese verkürzte Form wird auch als die kurze Weierstrassgleichung bezeichnet:

$$y^2 = x^3 + ax + b, \quad mit\ a,b \in\ K. \tag{2}$$

Die Glattheit von Kurven, die durch eine Weierstrassgleichung gegeben sind, lässt sich durch eine einfache Berechnung der Diskriminante überprüfen. Für Kurven, die in der kurzen Weierstrassgleichung gegeben sind hat die Diskriminante folgende Gleichung:

$$-\left(4a^3 + 27b^2\right).$$

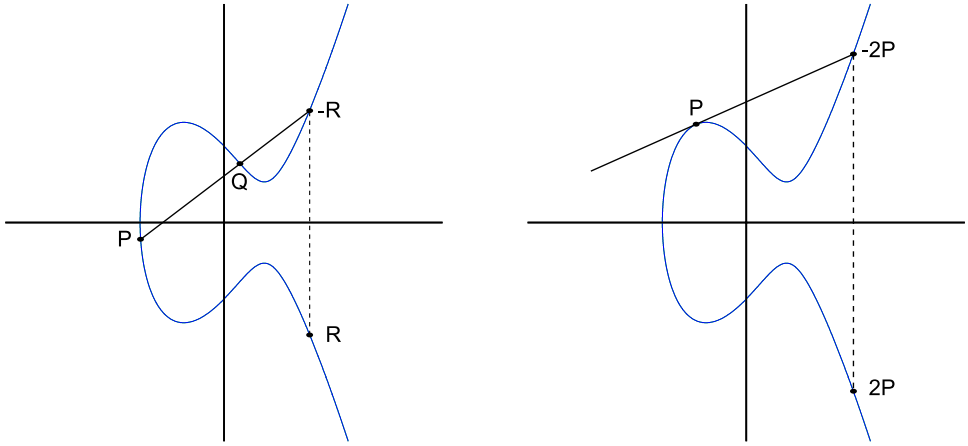
Ist die Diskriminante für eine Kurve ungleich Null, so ist diese glatt und hat keine Singularitäten [Kob98].

2.2. Addition von Kurvenpunkten einer elliptischen Kurve

Elliptische Kurven können über beliebige Körper K definiert werden. Für die Punktemenge $E(K)$ einer elliptischen Kurve lässt sich eine Gruppenstruktur mit der Addition von Punkten angeben. Der Einfachheit halber wird die Addition von Kurvenpunkten an elliptischen Kurven über den reellen Zahlen, die in der kurzen Weierstrassgleichung (2) geben sind, erklärt. Für beliebige Körper und auch für Kurven in der allgemeinen Weierstrassgleichung (1) kann die Punktaddition angegeben werden, wobei das Grundprinzip bestehen bleibt aber kleine Unterschiede existieren [Kob87a].

2.2.1. Punktaddition

Die Addition von zwei Punkten P und Q aus $E(\mathbb{R})$ lässt sich grafisch beschreiben, dazu wird eine Gerade durch beide Punkte gelegt. Diese Gerade schneidet die Kurve in einem weiteren Punkt $-R$, wird dieser an der X-Achse gespiegelt erhält man den Punkt R, für den $P + Q = R$ gilt. Durch eine Spiegelung an der X-Achse wird zu einem Punkt $R = (x_R,y_R)$ gerade der negative Punkt $-R = (x_R, -y_R)$ bestimmt. Kurven, die in der kurzen Weierstrassgleichung (2) gegeben sind, sind symmetrisch bezüglich der X-Achse [Hus03]. Abbildung 2(a) zeigt die Punktaddition zweier Punkte einer elliptischen Kurve über dem Körper der reellen Zahlen mit der Gleichung $y^2 = x^3 - 5x + 6$.



(a) Punktaddition (b) Punktverdopplung

Abbildung 2: Punktaddition von Punkten einer elliptischen Kurve

Mit Hilfe der Geradengleichung lässt sich die Punktaddition der Punkte $P = (x_P,y_P)$ und $Q = (x_Q,y_P)$ auch in expliziten Formeln angeben:

$$\begin{aligned} P + Q = R &= (x_R,y_R) \\ x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - x_R) - y_P \\ \lambda &= \frac{y_Q - y_P}{x_Q - x_P} \end{aligned} \tag{3}$$

Die Formeln der Punktaddition in (3) sind nur gültig, falls $char(K) \neq 2, 3$ und für die Punkte P sowie Q folgendes gilt: $P \neq \pm Q$. Die X-Koordinaten der Summanden dürfen demzufolge nicht gleich sein, da sonst durch Null dividiert würde. Für elliptische Kurven über Körper mit der Charakteristik von zwei oder drei lassen sich entsprechende explizite Formeln angeben.

2.2.2. Punktverdopplung

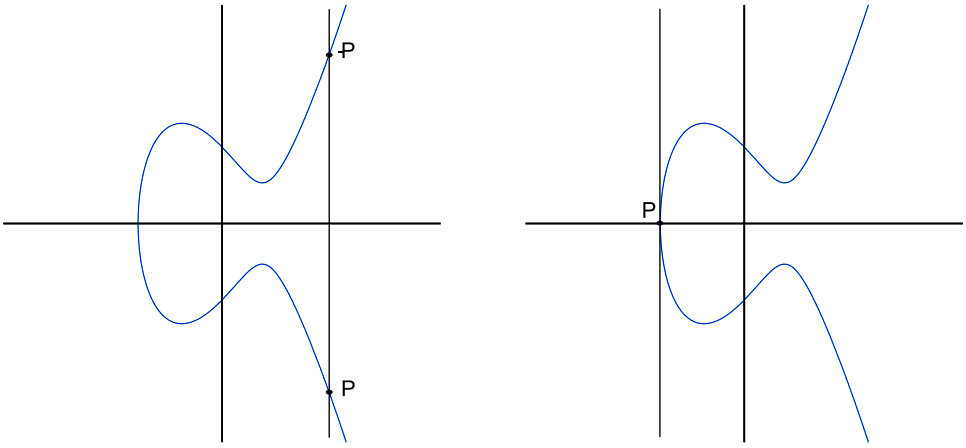
Sollen die Punkte P und Q mit $P = Q$ addiert werden, spricht man von einer Punktverdopplung. In diesem Fall wird im Punkt P eine Tangente an die elliptische Kurve angelegt, welche die Kurve in einem weiteren Punkt schneidet. Dieser Schnittpunkt wird wiederum an der X-Achse gespiegelt und man erhält den Ergebnispunkt $2P = P + P$. Abbildung 2(b) veranschaulicht diesen Fall. Die Punktverdopplung lässt sich mit Hilfe der Tangentengleichung abermals als explizite Formeln angeben ($char(K) \neq 2, 3$):

$$\begin{aligned} P + P = 2P &= (x_R,y_R) \\ x_R &= \lambda^2 - 2x_P \\ y_R &= \lambda(x_P - x_R) - y_P \\ \lambda &= \frac{3x_P^2 + a}{2y_P} \end{aligned} \tag{4}$$

Die Addition eines Punktes P mit seinem Inversen $Q = (-P)$ ergibt laut Definition das neutrale Element der Gruppe, den unendlich fernen Punkt $\mathcal{O} := (\infty,\infty)$. Die Geraden durch zueinander inversen Punkten verlaufen immer parallel zur Y-Achse und können somit keinen dritten Punkt auf der Kurve schneiden. Daher wird der unendlich ferne Punkt $\mathcal{O}$, welcher nicht auf der Kurve liegt, hinzugenommen. Abbildung 3(a) verdeutlicht die Addition der Punkte $P = (x_P,y_P)$ und $-P = (x_P, -y_P)$ mit $\mathcal{O}$ als Ergebnis. Man kann sich den Punkt $\mathcal{O}$ als einen Punkt vorstellen, der auf jeder senkrechten Geraden unendlich weit weg in Richtung der positiven Y-Werte liegt. Die Darstellung 3(b) zeigt einen zweiten Sonderfall, bei dem die Y-Koordinate des Punktes $P = (x_P,0)$ gleich Null ist. Die Verdopplung von P ergibt hierbei $P + P = \mathcal{O}$.

Demnach existieren für eine Addition von zwei Punkten einer elliptischen Kurve zwei unterschiedliche Berechnungsarten. Werden zwei unterschiedliche Punkte miteinander addiert, wird die Summe mittels der Punktaddition (ECADD) berechnet, bei der Addition von zwei gleichen Punkten kommt die Punktverdopplung (ECDBL)zum Einsatz. Die Punktemenge $E(K)$ bildet zusammen mit der Addition von Kurvenpunkten eine abelsche Gruppe in der folgendes gilt:

- $P + \mathcal{O} = P$ für alle $P \in E(K)$. $\mathcal{O}$ ist das neutrale Element der Gruppe.
- $P + Q = Q + P$ für alle $P,Q \in E(K)$. Die Punktaddition ist assoziativ.
- Für alle $P \in E(K)$ gibt es einen inversen Punkt $-P$ mit $P + (-P) = \mathcal{O}$.
- $(P + Q) + R = P + (Q + R)$ für alle $P,Q,R \in E(K)$. Die Punktaddition ist kommutativ.



(a) Punktaddition von P und -P (b) Punktverdopplung von $P = (x_P,0)$

Abbildung 3: Sonderfälle bei der Punktaddition von Punkten einer elliptischen Kurve

Die Beweise für die expliziten Formeln der Punktaddition und für die Feststellung, dass die Punktaddition $E(K)$ zu einer abelschen Gruppe macht sind in [Sil85] nachzulesen.

2.3. Elliptische Kurven über Primkörper

Die Addition von Punkten bei elliptischen Kurven über $\mathbb{R}$ ist langsam und zudem ungenau. Computer können reelle Zahlen nur mit einer bestimmten Genauigkeit darstellen, wodurch Rundungsfehler auftreten können. Bei der Verschlüsselung eines Textes würden Rundungsfehler dazu führen, dass der wieder entschlüsselte Text Unterschiede zum Originaltext aufweist. Werden die elliptischen Kurven aber über endliche Körper definiert umgeht man diese Probleme. Die Arithmetik bei endlichen Körpern lässt sich effizient implementieren und Rundungsfehler können nicht auftreten. Für Kryptosysteme sind vor allem elliptische Kurven über den Primkörpern $\mathbb{F}_p$ interessant [Eng99].

Definition. *Sei $p > 3$ eine Primzahl. Eine elliptische Kurve über einem Primkörper $\mathbb{F}_p$ ist die Menge aller Lösungen $(x,y) \in \mathbb{F}_p \times \mathbb{F}_p$ zur Kongruenz*

$$y^2 \equiv x^3 + ax + b \mod p$$

zusammen mit $\mathcal{O}$, dem Punkt im Unendlichen. Für die Konstanten $a,b \in \mathbb{F}_p$ muss zudem

$$4a^3 + 27b^2 \not\equiv 0 \mod p$$

gelten [Sti02].

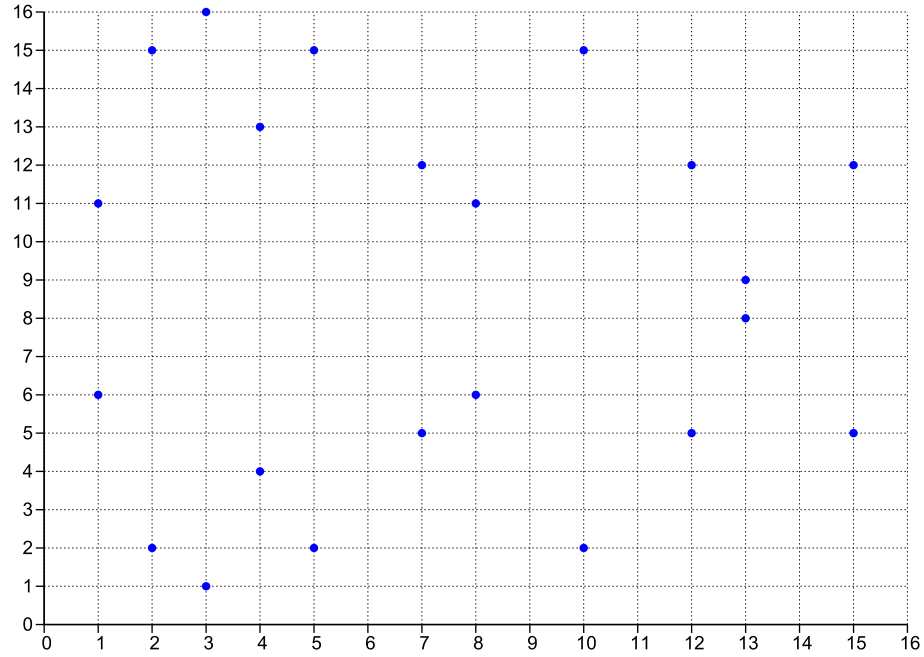


Abbildung 4: Beispiel einer elliptischen Kurve über $\mathbb{F}_p$

Aufgrund der Endlichkeit des Primkörpers $\mathbb{F}_p$ sind nur endlich viele Möglichkeiten für die X- und Y-Koordinaten der Kurvenpunkte vorhanden. Aus diesem Grund bildet $E(\mathbb{F}_p)$ eine endliche Abelsche Gruppe. Der Graph einer elliptischen Kurve über einem Primkörper besteht aus einzelnen Punkten und ist keine zusammenhängende Kurve. Abbildung 4 zeigt den Graph der elliptischen Kurve $y^2 = x^3 - 5x + 6$ über dem Körper $\mathbb{F}_{17}$ [Eng99].

Beispiel 1. *Sei* $E : y^2 = x^3 - 5x + 6$ *eine elliptische Kurve über* $\mathbb{F}_{17}$.

$$E(\mathbb{F}_{17}) = \{(1,6),(1,11),(2,2),(2,15),(3,1),(3,16),(4,4),(4,13),(5,2),(5,15),$$
$$(7,5),(7,12),(8,6),(8,11),(10,2),(10,15),(12,5),(12,12),(13,8),(13,9),$$
$$(15,5),(15,12),\mathcal{O}\}$$

Die Kurve hat demnach insgesamt 23 Punkte.

Auch bei den elliptischen Kurven über Primkörper wird durch eine Spiegelung an der X-Achse zu einem Punkt $R = (x_R, y_R)$ gerade der negative Punkt $-R$ bestimmt, wobei in diesem Fall der negative y-Wert mit modulo p berechnet wird: $-R = (x_R, -y_R \mod p)$. Im Graph 4 sind nur die Positiven Klassenvertreter eingezeichnet, dadurch ergibt sich eine Symmetrieachse bei $y = 8{,}5$. Für jeden x-Wert eines Kurvenpunktes existieren folglich wiederum zwei y-Werte.

Beispiel 2. *Sei* $E : y^2 = x^3 - 5x + 6$ *eine elliptische Kurve über* $\mathbb{F}_{17}$ *und* $P \in E(\mathbb{F}_{17})$.

$$P = (1,6) \qquad -P = (1, -6 \mod 17) = (1,11)$$

Obwohl sich die Punktaddition bei elliptischen Kurven über Primkörper nicht so anschaulich beschreiben lässt, wie bei den elliptischen Kurven über den reellen Zahlen, behalten die expliziten Formeln (3) und (4) dennoch ihre Gültigkeit.

Beispiel 3. *Sei* $E : y^2 = x^3 - 5x + 6$ *eine elliptische Kurve über* $\mathbb{F}_{17}$ *und* $P,Q \in E(\mathbb{F}_{17})$.

$$R = P + Q = (2,15) + (8,11)$$
$$\lambda = \frac{11 - 15}{8 - 2} \equiv -4 \cdot 6^{-1} \equiv -4 \cdot 3 \equiv 5 \mod 17$$
$$x_R = 5^2 - 2 - 8 \equiv 15 \mod 17 \qquad y_R = 5 \cdot (2 - 17) - 15 \equiv 12 \mod 17$$
$$R = (15,12)$$

Für elliptische Kurven über einem endlichen Körper $\mathbb{F}$ mit q Elementen lässt sich die Anzahl der Kurvenpunkte $\#E(\mathbb{F})$ mit Hilfe des Theorems von Hasse abschätzen [Sil85]:

$$-2\sqrt{q} + q + 1 \leq \#E(\mathbb{F}_q) \leq 2\sqrt{q} + q + 1.$$

Die Menge der Kurvenpunkte hat demgemäß eine untere und obere Grenze, die von der Anzahl der Elemente des zugrunde liegenden Körpers $\mathbb{F}_q$ abhängig ist. Des weiteren ist $2\sqrt{q}$ im Vergleich zu q relativ klein, aus diesem Grund kann man die Kurvenpunkteanzahl grob mit $\#E(\mathbb{F}_q) \approx q$ angeben. Die Bestimmung der genauen Anzahl kann mit dem Punktezählalgorithmus von Schoof erfolgen sowie mit einem der wesentlich schnelleren Verfahren, dem Schoof-Elkies-Aktin Verfahren (SEA) oder dem Satoh-Algorithmus [HMV03].

2.4. Die skalare Punktmultiplikation

Eine k-malige Addition eines Punktes $P \in E(K)$ auf sich selbst

$$k \cdot P = \underbrace{P + P + \ \ldots \ + P}_{k-mal}$$

wird skalare Punktmultiplikation genannt. Für negative k wird gerade der negative Punkt $-P$ k-mal aufaddiert und eine Punktmultiplikation mit einem Skalar von Null ergibt schlichtweg das neutrale Element $\mathcal{O} = 0 \cdot P$ [CFA+06].

Die Multiplikation eines Skalars mit einem Kurvenpunkt ist die Grundvorraussetzung für Kryptosysteme mit elliptischen Kurven, wobei Kryptosysteme mit Signaturen eine leicht abgewandelte Form der skalaren Punktmultiplikation benötigen. Für eine Signaturverifikation muss für zwei Skalare k,l und zwei Kurvenpunkte $P,Q \in E(K)$

$$k \cdot P + l \cdot Q$$

berechnet werden. Somit werden zwei skalare Punktmultiplikationen miteinander addiert, diese Operation wird als multi-skalar Punktmultiplikation oder eine mehrfache Punktmultiplikation bezeichnet. Im allgemeinen Fall ist die mehrfache Punktmultiplikation eine Summe von m skalaren Multiplikationen

$$\sum_{j=1}^{m} k_j \cdot P_j,$$

wobei k_j die Skalare darstellen und P_j Punkte auf einer elliptischen Kurve sind [HMV03].

2.4.1. Lim-Lee-Kämmung

Mit Hilfe der Lim-Lee-Kämmung (Lim-Lee Combing) kann jede Punktmultiplikation mit einem Skalar in eine mehrfache Punktmultiplikation überführt werden. Lim und Lee haben dieses Verfahren für allgemeine Gruppen in [LL94] vorgestellt. Bei der Lim-Lee-Kämmung wird ausgenutzt, dass ein Skalar k mit der Bitlänge b in zwei Teilskalare k_1, k_2 der Bitlänge $b/2$ zerlegen werden kann, so dass $k = k_1 \cdot 2^{b/2} + k_2$ gilt. Eine Punktmultiplikation kP kann somit in $kP = k_1 2^{b/2} \cdot P + k_2 \cdot P$ umgeformt werden. Wird nun $P_1 = 2^{b/2}P$ und $P_2 = P$ gesetzt, dann ergibt sich aus $k_1P_1 + k_2P_2 = kP$ eine mehrfache Punktmultiplikation. Um den Punkt P_1 zu erhalten wird für P $b/2$-mal eine Punktverdopplung (ECDBL) ausgeführt.

Im Allgemeinen wird bei der Lim-Lee-Kämmung der Skalar k mit der Bitlänge b in n Teile gleicher Länge $a = b/n$ aufgeteilt:

$$k = k_{n-1}2^{a(n-1)} + k_{n-2}2^{a(n-2)} + \ \ldots \ + k_1 2^a + k_0.$$

Falls die Bitlänge b nicht durch n teilbar sein sollte, werden vor dem höchstwertigen Bit des Skalars so viele Nullen angehängt, bis dies der Fall ist. Die Umwandlung in eine mehrfache Punktmultiplikation findet dann auf folgender Weise statt:

$$\begin{aligned} k \cdot P &= k_{n-1} \underbrace{2^{a(n-1)} \cdot P}_{P_{n-1}} + k_{n-2} \underbrace{2^{a(n-2)} \cdot P}_{P_{n-2}} + \ \ldots \ + k_1 \underbrace{2^a \cdot P}_{P_1} + k_0 \cdot \underbrace{P}_{P_0} \\ &= k_{n-1} \cdot P_{n-1} + k_{n-2} \cdot P_{n-2} + \ \ldots \ + k_1 \cdot P_1 + k_0 \cdot P_0 \\ &= \sum_{j=1}^{n} k_j \cdot P_j \end{aligned}$$

Für die Berechnung der einzelnen P_j mit $j = 1, \ldots, n-1$ werden insgesamt $b - b/n$ Punktverdopplungen benötigt, wobei die gegebenenfalls angehängten Nullen zur Bitlänge b auch zählen [LL94].

2.5. Kryptographie mit elliptischen Kurven

Kryptosysteme mit elliptischen Kurven nutzen hauptsächlich die Tatsache aus, dass eine Punktmultiplikation $k \cdot P = Q$ für einen Kurvenpunkt P und einem Skalar k einfach zu berechnen ist, es aber im Allgemeinen sehr schwer ist, für gegebene Punkte P und Q solch eine Zahl k zu bestimmen. Das inverse Problem einer Punktmultiplikation, ein passendes k für zwei Kurvenpunkte zu finden, wird das Diskrete Logarithmusproblem für elliptische Kurven genannt. Das Diskrete Logarithmusproblem lässt sich für beliebige Gruppen definieren, aber nicht in jeder Gruppe ist dieses Problem gleich schwierig. Bis in die späten Achtziger Jahre war für die Sicherheit von den meisten Kryptosystemen die Schwierigkeit des diskreten Logarithmusproblems in der primen Restklassengruppe $(\mathbb{F}_p)^*$ von zentraler Bedeutung. Miller [Mil86] und Koblitz [Kob87b] schlugen die Punktegruppen $E(\mathbb{F}_p)$ der elliptischen Kurven für das Diskrete Logarithmusproblem vor. Bislang sind nur Algorithmen bekannt, die einen exponentiellen Aufwand haben, um das diskrete Logarithmusproblem für elliptische Kurvenpunkte zu lösen. Allerdings existieren für die primen Restklassengruppen subexponentielle Algorithmen. Aus diesem Grund können Kryptosysteme mit elliptischen Kurven bei einer kürzeren Schlüssellänge die gleiche Sicherheit aufweisen wie Kryptosysteme mit primen Restklassengruppen [MP98].

2.5.1. Das Problem des diskreten Logarithmus für elliptische Kurven

Für einen Punkt P aus der Punktegruppe $E(\mathbb{F}_p)$ ist dessen Ordnung die kleinste natürliche Zahl n für die $n \cdot P = \mathcal{O}$ gilt. Die Ordnung eines Punktes P gibt demnach an, wie oft der Punkt aufaddiert werden muss, um das neutrale Element $\mathcal{O}$ der Gruppe von Kurvenpunkten zu erhalten. Hat P die Ordnung n, so erzeugt dieser eine zyklische Untergruppe von $E(\mathbb{F}_p)$ mit n Elementen:

$$\langle P \rangle = \{P,2P,3P, \ \ldots \ ,(n-1)P,\mathcal{O}\}\,.$$

Die vom Kurvenpunkt P gebildete zyklische Untergruppe wird mit $\langle P \rangle$ angegeben und P wird der Erzeuger dieser Untergruppe genannt. Ist die Ordnung des Erzeugers P eine Primzahl p, so haben nach dem Satz von Lagrange alle Elemente der Untergruppe $\langle P \rangle$ die Ordnung p, mit Ausnahme des neutralen Elements $\mathcal{O}$. Der Satz von Lagrange besagt nämlich, dass die Ordnung jeder Untergruppe einer endlichen Gruppe G gerade die Ordnung der Gruppe G teilt. Hat nun die Gruppe $\langle P \rangle$ eine prime Gruppenordnung, kann diese keine weitere Untergruppe als sich selbst haben, somit sind alle Elemente, außer das neutrale Element, Erzeuger der Gruppe mit Ordnung p [Buc03].

Definition. *Gegeben sei ein Punkt $P \in E(\mathbb{F}_p)$ der Ordnung n und ein Punkt $Q \in \langle P \rangle$. Das elliptische Kurven diskrete Logarithmusproblem (ECDLP) ist ein $k \in [0,n-1]$ zu finden, für welches*

$$k \cdot P = Q \mod p \tag{5}$$

gilt. Die Zahl k wird der diskrete Logarithmus des Punktes Q zum Basispunkt P genannt.

Das diskrete Logarithmusproblem bei elliptischen Kurven entspricht somit der Aufgabe, für zwei gegebenen Punkten P und Q aus der Punktegruppe $E(\mathbb{F}_p)$ eine Zahl k zu finden, für die $k \cdot P = Q \mod p$ gilt, falls solch eine Zahl existiert [HMV03].

Möchte man das Problem des diskreten Logarithmus für elliptische Kurven lösen gibt es verschiedene Möglichkeiten. Die einfachste Methode wäre eine naive Suche, bei der man alle Zahlen von 0 bis $n-1$ solange in die Gleichung (5) einsetzt, bis diese erfüllt ist. Im ungünstigsten Fall müssen alle n Möglichkeiten probiert werden, somit hat die naive Suche einen Aufwand von der Größenordnung $O(n)$. Dieses Verfahren ist bei einer hinreichend großen Ordnung n nicht praktikabel, für $n \geq 2^{80} \approx 1{,}2 \cdot 10^{24}$ ist der benötigte Berechnungsaufwand nicht mehr durchführbar. Eine wesentlich schnellere Methode ist der Babystep-Giantstep-Algorithmus von Shanks [Sha71] mit einem Berechnungsaufwand von $O(\sqrt{n})$. Für $n \geq 2^{160}$ ist aber auch dieses Verfahren in der Praxis nicht mehr einsetzbar, zudem benötigt der Babystep-Giantstep-Algorithmus einen Speicherplatzbedarf von $O(\sqrt{n})$. Der Pollard-ρ-Algorithmus von Pollard [Pol78] benötigt bei gleichem Aufwand $O(\sqrt{n})$ viel weniger Speicherplatz. Ferner kann der Pollard-ρ-Algorithmus parallelisiert werden, aus diesem Grund ist es möglich, durch parallele Berechnungen von m Prozessoren den Aufwand um den Faktor m auf $O(\sqrt{\frac{n}{m}})$ zu reduzieren. Eine weitere Methode zur Berechnung des ECDLP ist der Pohlig-Hellman-Algorithmus [PH78], welcher durch die Primfaktorzerlegung der Ordnung n das Problem auf die Untergruppen von $\langle P \rangle$ reduziert. Der diskrete Logarithmus muss dann nicht mehr direkt in $\langle P \rangle$ berechnet werden, sondern kann in dessen Untergruppen mit primer Ordnung bestimmt werden. Der Aufwand ist dann hauptsächlich von der größten Ordnung p der Untergruppen abhängig. Ist p der größte Primteiler von n, dann ist der Berechnungsaufwand für den Pohlig-Hellman-Algorithmus $O(\sqrt{p})$. Damit dieses Verfahren zur Bestimmung des ECDLP nicht angewendet werden kann, muss die Ordnung n einen sehr großen Primteiler $p \geq 2^{160}$ haben [HMV03].

Die vorgestellten Verfahren können für beliebige elliptische Kurven zur Bestimmung des ECDLP eingesetzt werden, für spezielle Klassen von elliptischen Kurven existieren aber effizientere Verfahren. Diese Klassen von Kurven sind kryptographisch gesehen schwache Kurven und müssen vermieden werden, damit die effizienteren Algorithmen nicht angewendet werden können. Standardisierte kryptographisch starke Kurven werden in [Loc05] von der ECC Brainpool Arbeitsgruppe vorgeschlagen.

2.5.2. Domain Parameter für Elliptische Kurven

Grundvoraussetzung für die Verwendung der public-key-Kryptosysteme mit elliptischen Kurven ist eine eindeutige Definition der genutzten Kurve für alle teilnehmenden Par-

teien. Eine elliptische Kurve wird eindeutig durch ihre Domain Parameter beschrieben, welche den zugrunde liegenden Körper, die Konstanten der Weierstrassgleichung, einen Erzeuger sowie dessen Ordnung enthalten. Im Einzelnen bestehen die Domain Parameter einer Kurve aus:

p : Die Primzahl p spezifiziert den Körper $\mathbb{F}_p$.

a,b : Die Konstanten a,$b \in \mathbb{F}_p$ legen die Weierstrassgleichung fest.

P : Ein Punkt auf der elliptischen Kurve, der ein Erzeuger einer Untergruppe von $E(\mathbb{F}_p)$ ist.

n : Die Ordnung des Punktes P ist n. n muss eine Primzahl sein.

Beispiel 4 (EC-Domain Parameter)**.** *Sei* $E : y^2 = x^3 + 2x + 4$ *eine elliptische Kurve über* $\mathbb{F}_{23}$.

$$E(\mathbb{F}_{23}) = \{(0,2),(0,21),(2,4),(2,19),(5,1),(5,22),(6,5),(6,18),(7,4),(7,19),(8,7),\\ (8,16),(10,9),(10,14),(11,0),(12,10),(12,13),(14,4),(14,19),(17,11),\\ (17,12),(19,1),(19,22),(22,1),(22,22),\mathcal{O}\}$$

Die Kurve hat demnach insgesamt 26 Punkte. Der Punkt $P = (17,12)$ *erzeugt folgende Untergruppe:*

$$\langle P\rangle = \{1P,2P,3P, \ \dots \ ,(n-1)P,\mathcal{O}\} = \{(17,12),(14,9),(0,2),(19,22),(12,13),\\ (6,18),(6,5),(12,0),(19,1),(0,21),(14,4),(17,11),\mathcal{O}\}.$$

Der Punkt P *hat somit die Ordnung* $n = 13$ *und* n *ist eine Primzahl. Damit ergeben sich die folgenden Domain Parameter:*

$$D = (p,a,b,P,n) = (23,2,4,(17,12),13).$$

Der private Schlüssel ist dann eine zufällige ganze Zahl $s \in [1,n-1]$ und der öffentliche Schlüssel gerade das Ergebnis der Punktmultiplikation von s und dem Punkt P. Möchte ein Angreifer aus dem öffentlichen Schlüssel den privaten berechnen, muss er das EC diskrete Logarithmusproblem lösen. Dazu kann er die oben genannten Algorithmen anwenden. Diese haben aber alle einen exponentiellen Aufwand, dementsprechend reicht ein s in der Größenordnung von 2^{160} aus, um dies dem Angreifer unmöglich zu machen. Anders ausgedrückt muss der geheime Schlüssel s eine Schlüssellänge von mindestens 160 Bits aufweisen, damit dieser nicht aus dem öffentlichen Schlüssel in praktikabler Zeit berechnet werden kann. Die Schlüssellänge ist somit ein Maß für die Sicherheit des privaten Schlüssels. Für die prime Restklassengruppe $(\mathbb{F}_p)^*$ existiert der sub-exponentielle Index-Calculus Algorithmus [Odl85]. Aus diesem Grund muss die Schlüssellänge bei Kryptosystemen mit dieser Gruppe wesentlich länger sein. Auch das

Faktorisierungsproblem, große Zahlen in das Produkt von Primzahlen zu zerlegen, lässt sich mit Hilfe des Zählkörpersiebs (number field sieve) mit einem sub-exponentiellem Aufwand berechnen [LL93]. Das europäische Forschungsnetzwerk Ecrypt setzt das Sicherheitsniveau in [GN06] eines 160-Bit EC-Schlüssels daher mit einem 1248-Bit langen Schlüssel für das RSA-Verfahren oder dem Digitalen Signaturen Algorithmus (DSA) gleich. Das RSA-Verfahren basiert auf dem Faktorisierungsproblem, wobei der Digitale Signaturen Algorithmus auf dem Diskreten Logarithmusproblem für die prime Restklassengruppe beruht. Die Firma Certicom hat 1997 zu einem öffentlichen Wettbewerb aufgerufen, in dem versucht werden soll, das EC-DLP für verschieden Schlüssellängen zu lösen. Dieser ECC-Wettbewerb liefert konkrete Hinweise für die tatsächliche Berechnungskomplexität des DLP für elliptische Kurven und ermöglicht den direkten Vergleich zwischen dem tatsächlichen Berechnungsaufwand und der Abschätzung der Aufwandsformeln in der O-Notation [BBF02]. Im November 2002 hat Chris Monico und sein Team von der Notre Dame Universität in Indiana den EC diskreten Logarithmus für einen 109 Bit langen Schlüssel bestimmt. Für die Lösung rechneten 10 000 Computer 24 Stunden am Tag und brauchten dazu genau 549 Tage. Certicom schätzt die Berechnung eines 163 Bitschlüssels eine hundertmillionmal schwerer ein als einen Schlüssel mit nur 109 Bit. Die nächste Wettbewerbstufe liegt bei einer Schlüssellänge von 131 Bit und Certicom geht davon aus, dass die Berechnung mehrere tausendmal mehr Rechenleistung benötigt [Cer02].

2.5.3. Vorteile von EC-Kryptographie

In [WMPW98] vergleichen die Autoren die Laufzeiten einer Softwareimplementierung der elliptischen Kurvenversion von DSA (EC-DSA) mit dem RSA-Verfahren und der Primkörperversion von DSA. Für EC-DSA wurde eine Schlüssellänge von 191 Bit und für die beiden anderen Verfahren ein Schlüssel mit 1024 Bit verwendet, wobei das Sicherheitsniveau des EC-Schlüssels etwas höher ist. Die Messwerte in Tabelle 1 sind alle in Millisekunden angegeben und wurden auf einem Pentium Pro mit 200 Mhz bestimmt.

In [LRW03] wurden die Laufzeiten der Signaturoperationen von RSA mit der von EC-DSA für verschiedene Schlüssellängen miteinander verglichen. Die Messungen wurden auf dem Personal Digital Assistant (PDA) Sharp Zaurus SL-5500G durchgeführt. Nach [GN06] ist das Sicherheitsniveau der EC-Schlüssel in der Vergleichstabelle 2 etwas höher als die entsprechenden RSA-Schlüssel.

Operation des Signaturverfahren	EC-DSA mit $E(\mathbb{F}_p)$	RSA	DSA
Signaturerzeugung in ms	6,3	43,3	23,6
Signaturverifikation in ms	26	0,65	28,3

Tabelle 1: Laufzeitenvergleich der Signaturoperationen von EC-DSA, DSA und RSA auf einem Pentium Pro mit 200 MHz

Signaturverfahren	Schlüssellänge in Bits	Signaturerzeugung in ms	Signaturverifikation in ms
EC-DSA	163	5,7	17,9
RSA	1024	78,0	4,3
EC-DSA	193	7,6	26,0
RSA	1536	251,9	9,7
EC-DSA	233	10,1	37,3
RSA	2240	731,8	20,4

Tabelle 2: Vergleich der Ausführungszeiten der Signaturoperationen von EC-DSA mit $E(\mathbb{F}_{2^m})$ und RSA mit $(\mathbb{F}_p)^*$ auf einem Sharp Zaurus SL-5500G

Tabellen 1 und 2 zeigen, dass die Signaturverifikation für die elliptische Kurvenversion durchweg etwas länger dauert, wobei der Unterschied bei steigender Schlüssellänge geringer wird. Dagegen ist die EC-DSA Signaturerzeugung wesentlich schneller und der Geschwindigkeitsvorteil steigt je länger der Schlüssel wird. Für ein höheres Sicherheitsniveau reicht beim EC-Schlüssel eine Anhebung um wenige Bits, dagegen muss der entsprechende RSA-Schlüssel um einige Hundert Bits verlängert werden. Diese Tatsachen machen Kryptosysteme mit elliptischen Kurven als Alternative für die Kryptosysteme, deren Sicherheit auf dem Faktorisierungsproblem oder dem DLP in Primkörpern beruht, interessant. Des weiteren werden durch die wesentlich kürzeren Schlüssel deutlich weniger Speicherplatz und Rechenleistung benötigt. Dies macht den Einsatz von EC-Kryptosystemen besonders für rechenschwache Systeme, wie Smartcards und PDAs attraktiv [BBF02].

2.6. Kryptosysteme mit elliptischen Kurven

In diesem Abschnitt werden kryptographische Verfahren vorgestellt, deren Sicherheit darauf beruhen, dass das diskrete Logarithmusproblem für elliptische Kurven schwer zu lösen ist. Aus diesem Grund kommen für diese Verfahren nur kryptographisch starke elliptische Kurven mit geeigneten Domain Parametern in Frage. Ursprünglich wurden diese Verfahren für die prime Restklassengruppe entwickelt und erst später für die Gruppe der elliptischen Kurvenpunkte angepasst.

2.6.1. Schlüsselpaarerzeugung

Für EC-Kryptosysteme ist das Schlüsselpaar, bestehend aus dem öffenlichen Schlüssel Q und dem privaten Schlüssel s, mit den Domain Parametern der zugehörigen elliptischen Kurve verbunden. Ein EC-Schlüsselpaar wird wie folgt erzeugt [HMV03]:

Algorithmus 1 EC-Schlüsselpaarerzeugung

Benötigt: Domain Parameter $D = (p,a,b,P,n)$.
Liefert: öffentlicher Schlüssel Q, privater Schlüssel s.

1. Wähle zufällig eine ganze Zahl $s \in [1,n-1]$.
2. Berechne Q mit der Punktmultiplikation $s \cdot P = Q$.
3. **return** Das Schlüsselpaar (s,Q).

Beispiel 5 (EC-Schlüsselpaarerzeugung)**.** *Sei $E : y^2 = x^3 + 2x + 4$ die elliptische Kurve über $\mathbb{F}_{23}$ aus Beispiel 4 mit den Domain Parametern*

$$D = (p,a,b,P,n) = (23,2,4,(17,12),13).$$

Der private Schlüssel sei $s = 5 \in [1,12]$, zusammen mit P ergibt das den öffentlichen Schlüssel $Q = 5 \cdot P = 5 \cdot (17,12) = (12,13)$.

2.6.2. EC-Diffie-Hellman Schlüsselaustausch

Möchten zwei Parteien über einen unsicheren Kanal einen geheimen Schlüssel austauschen, können sie dafür das Diffie-Hellman Schlüsselaustauschverfahren für elliptische Kurven nutzen [DH76]. Mit dem Verfahren von Diffie und Hellman können zum Beispiel Alice und Bob einen Schlüssel über das Internet austauschen, den keine lauschende Person erfährt.

Algorithmus 2 EC-Diffie-Hellman Schlüsselaustausch (EC-DH)

Benötigt: Domain Parameter $D = (p,a,b,P,n)$.
Liefert: gemeinsamer geheimer Schlüssel Q

1. Alice wählt zufällig eine geheime ganze Zahl $k_a \in [1,n-1]$. Sie berechnet $k_a \cdot P$ und schickt das Ergebnis an Bob
2. Bob wählt zufällig eine geheime ganze Zahl $k_b \in [1,n-1]$. Er berechnet $k_b \cdot P$ und schickt das Ergebnis an Alice
3. Alice berechnet mit Hilfe des geheimen Wertes k_a und dem Ergebnis von Bob den gemeinsamen geheimen Punkt $Q = k_a \cdot (k_b P)$.
4. Bob berechnet mit Hilfe des geheimen Wertes k_b und dem Ergebnis von Alice den gemeinsamen geheimen Punkt $Q = k_b \cdot (k_a P)$.

Alice und Bob besitzen am Ende den gemeinsamen geheimen Punkt $Q = (k_a k_b) \cdot P$. Das Diffie-Hellman Problem für elliptische Kurven (EC-DHP) beschreibt die Schwierigkeit aus $k_a P$ und $k_b P$ den Punkt $Q = (k_a k_b) \cdot P$ zu bestimmen. Dieses Problem lässt sich leicht lösen, falls man diskrete Logarithmen für elliptische Kurven lösen kann. Es ist

aber nicht bekannt, ob beide Probleme äquivalent sind. Wäre ein Angreifer dazu in der Lage das EC-DLP zu lösen, müsste er nur die Kommunikation belauschen und könnte den diskreten Logarithmus k_a von k_aP berechnen. Danach erhält er wie Alice mittels der Nachricht von Bob den geheimen Punkt Q [BSS05].

Beispiel 6 (EC-Diffie-Hellman Schlüsselaustausch)**.** *Sei* $E : y^2 = x^3 + 2x + 4$ *und* $D = (p,a,b,P,n) = (23,2,4,(17,12),13)$.
Alice wählt $k_a = 5$ *berechnet* $Q_a = 5P = 5 \cdot (17,12) = (12,13)$ *und sendet* Q_a *an Bob.*
Bob wählt $k_b = 2$ *berechnet* $Q_b = 2P = 2 \cdot (17,12) = (14,9)$ *und sendet* Q_b *an Alice.*
Alice berechnet den gemeinsamen geheimen Punkt $Q = 5Q_b = 5 \cdot (14,9) = (0,21)$.
Bob berechnet den gemeinsamen geheimen Punkt $Q = 2Q_a = 2 \cdot (12,13) = (0,21)$.

2.6.3. EC-ElGamal Public-Key Verschlüsselung

Für einen Schlüsselaustausch mit dem Diffie-Hellman Verfahren ist es unabdingbar, dass beide Parteien zur gleichen Zeit einen gemeinsamen Schlüssel bestimmen möchten, da beide Parteien interaktiv miteinander agieren müssen. Möchte Alice zum Beispiel verschlüsselte E-Mails an Bob verschicken, wäre es unpraktisch, immer zuerst interaktiv mit Bob einen geheimen Schlüssel für die Verschlüsselung der einzelnen Mail bestimmen zu müssen. Ein Verfahren für dieses Szenario ist das ElGamal Verschlüsselungsverfahren [Gam85], bei dem öffentliche Schlüssel genutzt werden. Aus diesem Grund muss Bob zuerst ein EC-Schlüsselpaar (s,Q) mit Algorithmus 1 erstellen und den öffentlichen Schlüssel Alice mitteilen oder in einem Verzeichnis veröffentlichen. Der Einfachheit halber wird angenommen, dass die Nachricht $m \in E(\mathbb{F}_p)$ von Alice ein elliptischer Kurvenpunkt ist. In [Cer00] sind standardisierte Verfahren zu finden, die elliptische Kurvenpunkte in verschiedene Datentypen überführen. Somit ist es möglich, beliebige Nachrichten mit Punkten der elliptischen Kurve zu identifizieren.

Algorithmus 3 EC-ElGamal Public-Key Verschlüsselung

Benötigt: Domain Parameter $D = (p,a,b,P,n)$, öffentlicher Schlüssel Q von Empfänger und Nachricht $m \in E(\mathbb{F}_p)$.
Liefert: Verschlüsselte Nachricht (R_1,R_2).

1. Alice wählt zufällig eine geheime ganze Zahl $k \in [1,n-1]$.
2. Sie berechnet $R_1 = k \cdot P$.
3. Sie berechnet $R_2 = m + k \cdot Q$.
4. Alice sendet (R_1,R_2) an Bob.

Bob empfängt die verschlüsselte Nachricht (R_1,R_2) und kann diese mit Hilfe seines pri-

vaten Schlüssels s wieder entschlüsseln. Dazu muss er folgendes berechnen:

$$\begin{aligned} R_2 - s \cdot R_1 &= (m + k \cdot Q) - s(k \cdot P) \\ &= (m + k \cdot Q) - k(s \cdot P) = (m + k \cdot Q) - k \cdot Q \qquad \text{,da } Q = s \cdot P \\ &= m \end{aligned}$$

Ein Angreifer kennt die öffentlichen Domain Parameter und den öffentlichen Schlüssel des Empfängers, durch Belauschen der Kommunikation erfährt er zudem die verschlüsselte Nachricht. Falls der Angreifer das EC-DLP lösen kann, ist es ihm möglich, aus dem öffentlichen Schlüssel Q den privaten Schlüssel s zu bestimmen und die Nachricht zu entschlüsseln. Im Übrigen kann er auch den diskreten Logarithmus k von R_1 lösen und damit durch $m = R_2 - k \cdot Q$ die Nachricht entschlüsseln. Es ist keine Möglichkeit bekannt, mit der ein Angreifer die Nachricht entschlüsseln kann, ohne das EC-DLP zu lösen [Was03].

Beispiel 7 (EC-ElGamal Public-Key Verschlüsselung)**.** *Sei* $E : y^2 = x^3 + 2x + 4$, $D = (p,a,b,P,n) = (23,2,4,(17,12),13)$ *und Bobs Schlüsselpaar* $(s,Q) = (5,(12,13))$.
Alice wählt $k = 3$ *und berechnet* $R_1 = 3P = 3 \cdot (17,12) = (0,2)$.
Alice berechnet für die Nachricht $m = (7,19)$*:* $R_2 = (7,19)+3\cdot(12,13) = (7,19)+(14,19) = (2,4)$.
Alice sendet das Tupel $(R_1,R_2) = ((0,2),(2,4))$ *an Bob.*
Bob entschlüsselt mit $(2,4) - 5 \cdot (0,2) = (2,4) - (14,19) = (7,19)$.

2.6.4. Elliptische Kurven Digitale Signaturen Algorithmus

Das National Institute of Standards and Technology (NIST) hat 1991 den Digitalen Signaturen Algorithmus (DSA) vorgeschlagen und später auch zum Standard erklärt. Scott Vanstone hat 1992 in [RHAL92] empfohlen für DSA nicht die prime Restklassengruppe, sondern die Gruppe der elliptischen Kurvenpunkte zu nutzen (EC-DSA). Mit dem EC-DSA Signaturverfahren können zum einen Nachrichten digital signiert werden und zum anderen kann auch überprüft werden, ob die Signatur authentisch ist. Alice schickt zum Beispiel eine von ihr signierte E-Mail an Bob. Für die Erstellung der Signatur benötigt Alice ein EC-Schlüsselpaar, da der geheime Schlüssel in die Signaturberechnungen mit einfließt. Zudem wird für EC-DSA eine kryptographisch sichere Hashfunktion H benötigt, damit beliebig lange Nachrichten signiert werden können. Bei der Berechnung der Signatur fließt nämlich nicht die ganze Nachricht an sich mit ein, sondern nur der Hashwert der Nachricht.

Algorithmus 4 EC-DSA Signaturerzeugung

Benötigt: Domain Parameter $D = (p,a,b,P,n)$, privater Schlüssel s des Signierers und Nachricht m.

Liefert: Signatur (c,d) der Nachricht m.

1. Wähle zufällig eine geheime ganze Zahl $k \in [1,n-1]$.
2. Berechne $k \cdot P = (x,y)$.
3. Berechne für die x-Koordinate $c = x \mod n$. Falls $c = 0$ ist, starte erneut mit Punkt 1.
4. Berechne den Hashwert der Nachricht $H(m) = e$.
5. Berechne $d = k^{-1}(e + sc) \mod n$. Falls $d = 0$ ist, starte erneut mit Punkt 1.
6. **return** Signatur (c,d).

Anhand der Signatur und dem öffentlichen Schlüssel von Alice kann Bob überprüfen, ob die E-Mail wirklich von ihr stammt. Wurde die Nachricht m auf dem Kommunikationsweg von einem Angreifer verändert, ändert sich zwangsläufig auch der Hashwert der veränderten Nachricht m'. Da bei der Signaturerzeugung nicht der Hashwert $H(m') = e'$ verwendet wurde, wird die Signatur für die veränderte Nachricht bei der Verifikation als ungültig erkannt.

Algorithmus 5 EC-DSA Signaturverifikation

Benötigt: Domain Parameter $D = (p,a,b,P,n)$, öffentlicher Schlüssel Q des Signierers, Nachricht m und Signatur (c,d).

Liefert: Signatur (c,d) der Nachricht m gültig oder ungültig.

1. Überprüfe, dass $c,d \in [1,n-1]$. Falls nicht ist die Signatur ungültig.
2. Berechne den Hashwert der Nachricht $H(m) = e$.
3. Berechne $w = d^{-1} \mod n$.
4. Berechne $u_1 = ew \mod n$ und $u_2 = cw \mod n$.
5. Berechne den Punkt $X = u_1 \cdot P + u_2 \cdot Q$. Falls $X = \mathcal{O}$ ist die Signatur ungültig.
6. Berechne für die x-Koordinate von X: $v = x \mod n$.
7. **return** Falls $v = c$ gilt ist die Signatur gültig, ansonsten ist die Signatur ungültig.

Falls die Signatur (c,d) für die Nachricht m wirklich vom Signierer mit dem Schlüsselpaar (s,Q) stammt, muss die Kongruenz $d \equiv k^{-1}(e + sc) \mod n$ gelten. Durch Umstellung und Einsetzten erhält man aus dieser

$$k \equiv d^{-1}(e + sc) \equiv d^{-1}e + d^{-1}sc \equiv we + wcs \equiv u_1 + u_2 s \mod n.$$

Für den Punkt $X = u_1 P + u_2 Q$ bei der Verifikation der Signatur gilt somit

$$X = u_1 P + u_2 Q = (u_1 + u_2 s)P = kP$$

und daraus folgt zwangsläufig, dass $v = c$ gilt. Möchte ein Angreifer eine Signatur fälschen, benötigt er den privaten Schlüssel, da dieser in den Berechnungen der Signatur

enthalten ist. Diesen erhält ein Angreifer nur durch eine diskrete Logarithmusberechnung des öffentlichen Schlüssels, er muss daher in der Lage sein das EC-DLP zu lösen [JMV01].

Beispiel 8 (EC-DSA Signaturerzeugung und Signaturverifikation)**.** *Sei* $E : y^2 = x^3 + 2x+4$*,* $D = (p,a,b,P,n) = (23,2,4,(17,12),13)$ *und Alice Schlüsselpaar* $(s,Q) = (5,(12,13))$*.*
Alice signiert die Nachricht m ***für Bob mit ihrem privaten Schlüssel*** s***:***
Alice wählt $k = 4$ *und berechnet* $4P = 4 \cdot (17,12) = (19,22)$*.*
Alice berechnet $19 \mod 13 = 6$*, somit ist* $c = 6$*.*
Der Hashwert der Nachricht sei $H(m) = 8$*.*
Alice berechnet $d = 4^{-1}(8 + 5 \cdot 6) \mod 13 = 10(8 + 30) \mod 13 = 3$*.*
Alice hängt die Signatur $(c,d) = (6,3)$ *an die Nachricht.*

Bob verifiziert die Signatur $(c,d) = (6,3)$ ***der Nachricht*** m ***mit dem öffentlichen Schlüssel*** Q ***von Alice:***
$6 \in [1,12]$ *und* $3 \in [1,12]$*.*
Der Hashwert der Nachricht sei wieder $H(m) = 8$*.*
Bob berechnet $w = 3^{-1} \mod 13 = 9$*.*
Bob berechnet $u_1 = 8 \cdot 9 \mod 13 = 7$ *und* $u_2 = 6 \cdot 9 \mod 13 = 2$*.*
Bob berechnet den Punkt $X = 7P+2Q = 7\cdot(17,12)+2\cdot(12,13) = (6,5)+(0,21) = (19,22)$*.*
Bob berechnet für die x-Koordinate von X*:* $v = 19 \mod 13 = 6$*.*
Da $v = c$ *gilt, ist die Signatur* $(6,3)$ *gültig. Die Nachricht* m *ist demnach wirklich von Alice.*

3. Darstellungen ganzer Zahlen

In Kapitel 2.6 wurden einige Kryptosysteme mit elliptischen Kurven vorgestellt. Diese Systeme nutzen ganze Zahlen, die mit elliptischen Kurvenpunkten multipliziert werden. Ganze Zahlen lassen sich nicht nur als Dezimalzahlen zur Basis zehn, sondern auch in anderen Darstellungsarten darstellen. In Computern werden ganze Zahlen zum Beispiel als eine Folge von Nullen und Einsen in der so genannten Binärdarstellung zur Basis zwei verwendet. Jede natürliche Zahl größer Eins kann als Basis zur Darstellung ganzer Zahlen verwendet werden. In diesem Kapitel sollen nur Darstellungen zur Basis zwei betrachtet werden.

3.1. Binärdarstellung

In der Binärdarstellung werden ganze Zahlen als eine Ziffernfolge von Nullen und Einsen repräsentiert, der Stellenwert jeder Ziffer entspricht dabei einer zu Stelle passenden Zweierpotenz. Der Wert der Binärzahl ergibt sich durch die Addition der Ziffern, nachdem sie mit ihrem Stellenwert multipliziert worden sind.

Definition (Binärdarstellung)**.** *Die Folge* $(d_{n-1},\ldots,d_0)$ *heißt Binärdarstellung der Ganzzahl* d*, falls*

$$d = \sum_{i=0}^{n-1} d_i \cdot 2^i$$

und $d_i \in \{0,1\}$ *für alle* $i = 0,\ldots,n-1$ *gilt.*

Die Zahl n wird als die Bitlänge der Binärzahl bezeichnet und es gilt $n = \lfloor \log_2 d \rfloor + 1$. Die einzelnen Ziffern d_i werden Bits genannt, welches die Kurzform für binary digits ist. Mittels Algorithmus 6 lässt sich eine Dezimalzahl d in dessen Binärdarstellung überführen, dabei wird d solange durch Zwei mit Divisionsrest geteilt, bis Null herauskommt. Der Rest der Division ergibt dabei jeweils die einzelnen Bits der gesuchten Binärdarstellung [Buc03, Dah05].

Beispiel 9. *Für* $d = 21$ *ergeben sich mit Algorithmus 6 folgende Rechenschritte:*

$$\begin{array}{rcll} 21/2 & = & 10 & \textit{Rest } 1 \quad \rightarrow d_0 = 1 \\ 10/2 & = & 5 & \textit{Rest } 0 \quad \rightarrow d_1 = 0 \\ 5/2 & = & 2 & \textit{Rest } 1 \quad \rightarrow d_2 = 1 \\ 2/2 & = & 1 & \textit{Rest } 0 \quad \rightarrow d_3 = 0 \\ 1/2 & = & 0 & \textit{Rest } 1 \quad \rightarrow d_4 = 1 \end{array}$$

Algorithmus 6 Dezimalzahl in Binärzahl darstellen

Benötigt: Ganzzahl d als Dezimalzahl.
Liefert: Binärdarstellung $(d_{n-1},\ldots,d_0)$ der Ganzzahl d.

1. $n \leftarrow 0$
2. **while** $d \neq 0$ **do**
3. **if** $d \bmod 2 = 1$ **then** {Divisionsrest ist 1}
4. $d_n \leftarrow 1$
5. **else** {Divisionsrest ist 0}
6. $d_n \leftarrow 0$
7. **end if**
8. $d \leftarrow \lfloor d/2 \rfloor$
9. $n \leftarrow n + 1$
10. **end while**
11. **return** $(d_{n-1},\ldots,d_0)$.

Die Ganzzahl lässt sich somit durch die Binärzahl (10101) *mit einer Bitlänge von* $5 = \lfloor \log_2 21 \rfloor + 1$ *darstellen. Den Dezimalwert der Binärzahl erhält man durch* $2^4 + 2^2 + 2^0 = 21$.

3.2. Vorzeichenbehaftete Darstellungen zur Basis Zwei

Für die Binärdarstellung ist die Ziffernmenge auf die Ziffern Null und Eins beschränkt. Erweitert man diese Menge mit beliebigen Ziffern inklusive Vorzeichen erhält man die $\mathcal{D}$-Darstellung, eine verallgemeinerte Ganzzahlrepräsentation zur Basis Zwei.

Definition ($\mathcal{D}$-Darstellung)**.** *Die Folge* $(d_{n-1},\ldots,d_0)$ *heißt* $\mathcal{D}$*-Darstellung der Ganzzahl* d*, falls*

$$d = \sum_{i=0}^{n-1} d_i \cdot 2^i$$

und $d_i \in \mathcal{D}$ *für alle* $i = 0,\ldots,n-1$ *gilt.*

Die Menge $\mathcal{D}$ heißt Ziffernmenge und beinhaltet alle zulässigen Ziffern der Darstellung. Die Anzahl der Ziffern und somit die Ordnung von $\mathcal{D}$, wird durch $|\mathcal{D}|$ dargestellt. Die Darstellung wird eine vorzeichenbehaftete Binärdarstellung (signed binary representation) genannt, wenn die Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ ist. Falls die Menge $\mathcal{D}$ nur die Ziffern Null und Eins ohne Vorzeichen enthält, ergibt das gerade die Binärdarstellung. Für $\mathcal{D} = \{0, \pm 1, \ldots, \pm x\}$ wird von einer vorzeichenbehafteten Darstellung (signed representation) gesprochen. Im allgemeinen Fall verlieren $\mathcal{D}$-Darstellungen ihre Eindeutigkeit, es sind dann mehrere Darstellungen für den selben Dezimalwert möglich. Zum Beispiel

lässt sich die Zahl 21 in der signed binary representation durch $(11\bar{1}01)$ oder auch durch $(1\bar{1}1\bar{1})$ mit der selben Bitlänge darstellen, wobei $\bar{1} = -1$ bedeutet [CFA+06].

Im Folgenden wird eine Klasse einer $\mathcal{D}$-Darstellung mit $\mathcal{X}$ bezeichnet, ein Beispiel für solch eine Klasse ist die Binärdarstellung. Weitere Darstellungsklassen und Algorithmen um diese für Ganzzahlen zu erlangen werden in Kapitel 5 vorgestellt.

Beispiel 10. *Ein Beispiel für die Ganzzahl* 21 *in der vorzeichenbehafteten Binärdarstellung mit* $\mathcal{D} = \{0, \pm 1\}$ *ist* $(10\bar{1}0\bar{1}\bar{1}) = 2^5 - 2^3 - 2^1 - 2^0$. *In der vorzeichenbehafteten Darstellung mit* $\mathcal{D} = \{0, \pm 1, \pm 3\}$ *könnte* 21 *durch* $(1301) = 2^3 + 3 \cdot 2^2 + 2^0$ *dargestellt werden.*

3.3. Hamming-Gewicht von Darstellungen

Das Hamming-Gewicht (Hamming weight) und die Hamming-Distanz (Hamming density) sind Maße für die Unterschiedlichkeit von Zeichenfolgen, welche dazu verwendet werden können, die verschiedenen $\mathcal{D}$-Darstellungsklassen miteinander zu vergleichen. Für die folgenden Definitionen sei $\mathcal{X}$ eine Klasse der $\mathcal{D}$-Darstellungen.

Definition (Hamming-Gewicht)**.** *Sei* $r = (d_{n-1}, \ldots, d_0)$ *eine* $\mathcal{D}$*-Darstellung mit der Bitlänge* n*. Das Hamming-Gewicht (HW) von* r *ist die Anzahl der Ziffern ungleich Null in* r *und wird mit* $\mathcal{HW}(r)$ *bezeichnet.*

Setzt man das Hamming-Gewicht in Relation zur Bitlänge erhält man die Hamming-Dichte einer Darstellung.

Definition (Hamming-Dichte)**.** *Sei* $r = (d_{n-1}, \ldots, d_0)$ *eine* $\mathcal{D}$*-Darstellung mit der Bitlänge* n*. Die Hamming-Dichte (HD) von* r *ist die Anzahl der Ziffern ungleich Null in* r *geteilt durch die Bitlänge von* r*. Die Hammnig-Dichte wird mit* $\mathcal{HD}(r)$ *bezeichnet, es gilt* $\mathcal{HD}(r) = \mathcal{HW}(r)/n$.

Das Hamming-Gewicht und die Hamming-Dichte sind nur für festgelegte Bitfolgen definiert. Für zufällige Bitfolgen einer Darstellungsklasse kann deren Dichte nur geschätzt werden. Die Schätzung erfolgt durch die Angabe der durchschnittlichen Hammning-Dichte (average Hamming density).

Definition (Durchschnittliche Hamming-Dichte)**.** *Die durchschnittliche Hamming-Dichte (AHD) einer Klasse* $\mathcal{X}$ *der* $\mathcal{D}$*-Darstellungen ist die erwartete Hamming-Dichte für eine zufällige Ziffernfolge in* $\mathcal{X}$ *mit der Bitlänge von* $n \to \infty$ *und wird mit* $\mathcal{AHD}(\mathcal{X})$ *bezeichnet.*

Alle oben genannten Hamming-Maße sind nicht nur für einzelne Bitfolgen definiert, sondern auch für mehrere Folgen gemeinsam (joint). Für das joint Hamming-Gewicht

(joint Hamming weight) werden die einzelnen Bitfolgen untereinander in einer Matrix angeordnet und die Anzahl der Spalten der Matrix bestimmt, die keine Nullspalten sind.

Definition (Joint Hamming-Gewicht)**.** *Seien* $r_1 = (d^1_{n-1},\ldots,d^1_0),\ldots,r_k = (d^k_{n-1},\ldots,d^k_0)$ *k* $\mathcal{D}$*-Darstellungen mit Bitlänge n. Das joint Hamming-Gewicht (JHW) von* $r_1,\ldots,r_k$ *ergibt sich aus der Anzahl der Spalten in der Matrix*

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{pmatrix} = \begin{pmatrix} d^1_{n-1} & \ldots & d^1_0 \\ d^2_{n-1} & \ldots & d^2_0 \\ \vdots & & \vdots \\ d^k_{n-1} & \ldots & d^k_0 \end{pmatrix},$$

in denen mindestens eine Ziffer ungleich Null ist und wird mit $\mathcal{JHW}(r_1,\ldots,r_k)$ *bezeichnet. Die Bitlänge n wird durch die längste* $\mathcal{D}$*-Darstellung bestimmt, kürzere Darstellungen werden mit Nullen von links entsprechend aufgefüllt.*

Setzt man das gemeinsame Hamming-Gewicht in Relation zur maximalen Bitlänge n der betrachteten Darstellungen, so erhält man die joint Hamming-Dichte (joint Hamming density).

Definition (Joint Hamming-Dichte)**.**
Seien $r_1 = (d^1_{n-1},\ldots,d^1_0)$, …, $r_k = (d^k_{n-1},\ldots,d^k_0)$ *k* $\mathcal{D}$*-Darstellungen mit der maximalen Bitlänge n. Die gemeinsame Hamming-Dichte (JHD) von* $r_1,\ldots,r_k$ *ergibt sich durch* $\mathcal{JHD}(r_1,\ldots,r_k) := \mathcal{JHW}(r_1,\ldots,r_k)/n$.

Auch für mehrere zufällige Bitfolgen einer Darstellungsklasse lässt sich eine durchschnittliche gemeinsame Hamming-Dichte (average joint Hamming density) angeben.

Definition (Durchschnittliche joint Hamming-Dichte)**.** *Die durchschnittliche joint Hamming-Dichte (AJHD) einer Darstellungsklasse* $\mathcal{X}$ *ist die erwartete gemeinsame Hamming-Dichte von k zufälligen* $\mathcal{D}$*-Darstellungen in* $\mathcal{X}$ *mit der Bitlänge* $n \to \infty$ *und wird mit* $\mathcal{AJHD}_k(\mathcal{X})$ *bezeichnet.*

Beispiel 11. *Abschließend werden die obigen Definitionen an einem Beispiel mit den zwei Binärdarstellungen*

$$\begin{array}{lllllllllllll} r_1 & = & (1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0) \\ r_2 & = & & & & (1 & 0 & 0 & 0 & 1 & 0 & 1 & 0) \end{array}$$

mit $Bitlänge(r_1) = 11$ *und* $Bitlänge(r_2) = 8$ *verdeutlicht.*

$$\begin{array}{rr} \mathcal{HW}(r_1) = 5 & \mathcal{HD}(r_1) = 5/11 \\ \mathcal{HW}(r_2) = 3 & \mathcal{HD}(r_2) = 3/8 \\ \mathcal{JHW}(r_1,r_2) = 6 & \mathcal{JHD}(r_1,r_2) = 6/11 \end{array}$$

Für eine zufällige Bitfolge in der Binärdarstellung haben die Ziffern Null und Eins die gleiche Wahrscheinlichkeit von 1/2. Aus diesem Grund gilt für die durchschnittliche Hamming-Dichte:

$$\mathcal{AHD}(\text{Binärdarstellung}) = \frac{1}{2}.$$

Werden k Ganzzahlen gemeinsam betrachtet, ist die Wahrscheinlichkeit für eine Nullspalte $1/2^k$ und somit gilt

$$\mathcal{AJHD}_k(\text{Binärdarstellung}) = 1 - \frac{1}{2^k}$$

für die durchschnittliche joint Hamming-Dichte von k Binärdarstellungen [Dah05].

4. Evaluationsalgorithmen für die mehrfache Punktmultiplikation

Für die Ausführungszeiten von Kryptosystemen mit elliptischen Kurven sind die Berechnungszeiten der einfachen und mehrfachen Punktmultiplikationen entscheidend. Daher werden in diesem Kapitel Algorithmen betrachtet, mit denen skalare Punktmultiplikationen effizient berechnet werden können. Das Hauptaugenmerk ist dabei auf die mehrfache Punktmultiplikation gerichtet, die zum Beispiel bei Signaturverifikationen Verwendung findet.

Oftmals sind die Exponentiationsalgorithmen für multiplikativ geschriebene Gruppen einfach für die Nutzung der additiv geschriebenen Gruppe der elliptischen Kurvenpunkte angepasst worden. Aus diesem Grund wurde bei einigen Evaluationsalgorithmen für die Punktmultiplikation die Bezeichnung Exponentiation beibehalten, obwohl es sich um eine additiv geschriebene Gruppe handelt.

4.1. Binärer Exponentiationsalgorithmus

Die simpelste Methode für die Berechnung einer einfachen skalaren Punktmultiplikation $k \cdot P$ ist die k-malige Addition des Punktes P auf sich selbst, bei der genau eine Punktverdopplung und $k-2$ Punktadditionen benötigt werden. Eine wesentlich effizientere Methode ist der Binäre Exponentiationsalgorithmus für elliptische Kurvenpunkte, bei dem die n-Bit lange Binärdarstellung des Skalars

$$k = \sum_{i=0}^{n-1} d_i 2^i \quad mit\ d_i \in \{0,1\}$$

verwendet wird. Der Algorithmus 7 startet mit dem neutralen Element $R = \mathcal{O}$ und arbeitet den Skalar k Bit für Bit von links nach rechts ab. Für jedes Bit wird der Punkt R verdoppelt und falls das Bit ungleich Null ist, wird zusätzlich der Punkt P addiert.

Algorithmus 7 Binärer Exponentiationsalgorithmus

Benötigt: Skalar $k = (d_{n-1}d_{n-2}\ldots d_1 d_0)_2$ und Punkt $P \in E(\mathbb{F}_p)$.
Liefert: Skalare Punktmultiplikation $k \cdot P$.

1. $R \leftarrow \mathcal{O}$
2. **for** $i = n-1$ down to 0 **do**
3. $\quad R \leftarrow 2 \cdot R$ {ECDBL}
4. $\quad$ **if** $d_i = 1$ **then**
5. $\quad\quad R \leftarrow R + P$ {ECADD}
6. $\quad$ **end if**
7. **end for**
8. **return** R.

Diese Methode benötigt für einen n-Bit langen Skalar insgesamt $n-1$ Punktverdopplungen (ECDBL) und $HW(k)-1$ Punktadditionen (ECADD), wobei $HW(k)$ die Anzahl der Einsen in der Binärdarstellung des Skalars k ist und als Hamming-Gewicht bezeichnet wird. Die erste Punktverdopplung sowie die erste Punktaddition werden dabei nicht mitgezählt, da die Verdopplung von $\mathcal{O}$ wieder $\mathcal{O}$ ist und eine Addition von P und $\mathcal{O}$ gerade den Punkt P selbst ergibt. Im Durchschnitt besteht die Hälfte der Bits des Skalars k aus Einsen, daher verwendet der binäre Exponentiationsalgorithmus im Durchschnitt

$$n\ ECDBL + n \cdot \frac{1}{2}\ ECADD$$

Operationen für die Berechnung einer skalaren Punktmultiplikation $k \cdot P$. Algorithmus 7 heißt Exponentiationsalgorithmus, da er für multiplikative Gruppen entwickelt wurde und für die additive Gruppe $E(\mathbb{F}_p)$ angepasst wurde [Hen03, MvOV97].

Beispiel 12. *Die folgende Tabelle veranschaulicht die Berechnung der skalaren Punktmultiplikation $11 \cdot P$ für ein $P \in E(\mathbb{F}_p)$ mittels des Binären Exponentiationsalgorithmus. Die Binärdarstellung des Skalars ist $11 = (1011)_2$ mit der Bitlänge $n = 4$. Im Beispiel werden folglich insgesamt fünf Punktadditionen, genauer drei ECDBL- und zwei ECADD-Operationen benötigt. Wogegen die Methode des k-maligen Aufaddierens für die gleiche Punktmultiplikation insgesamt zehn Punktadditionen, eine ECDBL- und neun ECADD-Operationen, erfordern würde.*

i	d_i	ECDBL	ECADD	Variable R
				$R = \mathcal{O}$
3	1	$R \leftarrow \mathcal{O} + \mathcal{O}$	$R \leftarrow \mathcal{O} + P$	$R = 1P$
2	0	$R \leftarrow 1P + 1P$		$R = 2P$
1	1	$R \leftarrow 2P + 2P$	$R \leftarrow 4P + P$	$R = 5P$
0	1	$R \leftarrow 5P + 5P$	$R \leftarrow 10P + P$	$R = 11P$

Die binäre Exponentiationsmethode lässt sich auch für Skalare in einer $\mathcal{D}$-Darstellung anpassen, dazu müssen in Zeile 5 die Punkte $t \cdot P$, $t \in \mathcal{D}$ addiert werden. Damit die einzelnen Punkte $t \cdot P$ nicht bei jedem Auftreten neu berechnet werden müssen, werden diese zu Beginn der Methode in der so genannten Vorberechnungsphase ermittelt und gespeichert. Algorithmus 8 beschreibt den Exponentiationsalgorithmus für $\mathcal{D}$-Darstellungen. Der angepasste Algorithmus 8 führt nun für jede Ziffer d_i, die ungleich Null ist eine ECADD-Operation durch. Die Anzahl dieser durchgeführten Operationen hängt somit von der $\mathcal{AHD}$ der $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ ab. Damit ergibt sich für die Bestimmung einer Skalarmultiplikation $k \cdot P$ mit Skalar k in einer $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ ein durchschnittlicher Aufwand von

$$n\ ECDBL + n \cdot \mathcal{AHD}(\mathcal{X})\ ECADD$$

Operationen in der Hauptberechnungsphase. Der Aufwand der Vorberechnungsphase ist von der Darstellung des Skalars abhängig. Für jede Ziffer t aus $\mathcal{D}$ wird der Punkt $t \cdot P$

Algorithmus 8 Exponentiationsalgorithmus für $\mathcal{D}$-Darstellungen

Benötigt: Skalar $k = (d_{n-1}d_{n-2}\dots d_1 d_0)$ mit $d_i \in \mathcal{D}$ und Punkt $P \in E(\mathbb{F}_p)$.
Liefert: Skalare Punktmultiplikation $k \cdot P$.

1. {**Vorberechnung**}
2. $Q_t \leftarrow t \cdot P$, für alle $t \in \mathcal{D} \setminus \{0\}$
3. {**Hauptrechnung**}
4. $R \leftarrow \mathcal{O}$
5. **for** $i = n-1$ down to 0 **do**
6. $\quad R \leftarrow 2 \cdot R \qquad$ {ECDBL}
7. $\quad$ **if** $d_i \neq 0$ **then**
8. $\quad\quad R \leftarrow R + Q_{d_i} \qquad$ {ECADD}
9. $\quad$ **end if**
10. **end for**
11. **return** R.

ermittelt. Für t gleich Null und Eins ist keine Berechnung nötig. In der Vorberechnungsphase werden dementsprechend insgesamt $|\mathcal{D}| - 2$ Punkte vorberechnet, für die weitere Verdopplungs- und Additionsoperationen benötigt werden. Wird die $\mathcal{D}$-Darstellung geschickt gewählt, können trotz der zusätzlich benötigten Operationen in der Vorberechnungsphase, in der Hauptrechnung so viele Operationen eingespart werden, dass der Gesamtaufwand geringer ist als die Berechnung mittels dem binären Exponentiationsalgorithmus [Dah05]. Weitere effiziente Evaluationsalgorithmen für eine einfache skalare Punktmultiplikation sind in [Gor98] und [HMV03] zu finden.

Beispiel 13. *Die folgende Tabelle veranschaulicht die Berechnung der skalaren Punktmultiplikation* $13 \cdot P$ *für ein* $P \in E(\mathbb{F}_p)$ *mittels Algorithmus 8. Der Skalar* $13 = (301)$ *ist in der* $\mathcal{D}$*-Darstellung mit* $\mathcal{D} = \{0,1,3\}$ *gegeben.*

Vorberechnung	Hauptrechnung				
Q_t	i	d_i	ECDBL	ECADD	Variable R
$Q_1 \leftarrow P$					$R = \mathcal{O}$
$Q_3 \leftarrow 3P$	2	3	$R \leftarrow \mathcal{O} + \mathcal{O}$	$R \leftarrow \mathcal{O} + Q_3$	$R = 3P$
	1	0	$R \leftarrow 3P + 3P$		$R = 6P$
	0	1	$R \leftarrow 6P + 6P$	$R \leftarrow 12P + Q_1$	$R = 13P$

Mit dem Exponentiationsalgorithmus für $\mathcal{D}$-Darstellung lässt sich auch eine mehrfache Punktmultiplikation der Form $\sum_{j=1}^{m} k_j \cdot P_j$ berechnen. Hierfür werden die m Punktmultiplikationen separat ermittelt und die einzelnen Ergebnisse durch $(m-1)$ Punktadditionen zusammengezählt. Im Durchschnitt werden bei dieser Vorgehensweise exklusive der Vorberechnungen

$$n \cdot k\ ECDBL + (n \cdot k\ \mathcal{AHD}(\mathcal{X})) + (m-1)\, ECADD \tag{6}$$

Operationen benötigt. Die Skalare sind dabei in der $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ gegeben. Für die Vorberechnung der $m \cdot (|\mathcal{D}| - 2)$ Punkte fallen weitere ECADD- und ECDBL-Operationen an. In den folgenden Kapiteln werden effizientere Methoden für die mehrfache Punktmultiplikation vorgestellt [Dah05].

4.2. Interleave Methode

Eine mehrfache Punktmultiplikation der Form $k \cdot P + l \cdot Q$ lässt sich durch die Addition von zwei separaten einfachen Punktmultiplikationen berechnen. Wird jedoch diese mehrfache Punktmultiplikation nicht durch die drei Einzelschritte, sondern durch einen einzigen bestimmt, lassen sich einige Operationen einsparen. In [Möl01] wurde dazu die Interleave Methode vorgestellt, welche die ECDBL-Operationen für beide Skalare simultan und die ECADD-Operationen getrennt ausführt. Anschaulich gesehen werden die beiden Skalare in ihrer Binärdarstellung untereinander geschrieben

$$k = d_{n-1}\ d_{n-2}\ \cdots\ d_0$$
$$l = d_{n-1}\ d_{n-2}\ \cdots\ d_0$$

und von links nach rechts spaltenweise Bit für Bit abgearbeitet. Falls die Binärdarstellungen der Skalare unterschiedlich lang sind, werden der kürzeren Darstellung führende Nullen vorangestellt. Wie bei der Binärmethode wird mit dem neutralen Element $R = \mathcal{O}$ gestartet und für jede Bitspalte R verdoppelt. Zusätzlich wird in jeder Bitspalte überprüft, ob die Bits der Skalare ungleich Null sind. Ist in der Bitspalte i das Bit k_i ungleich Null wird der Punkt P addiert und sollte l_i ungleich Null sein wird der Punkt Q addiert. Falls in einer Spalte beide Bits gleichzeitig Eins sind, wird zuerst P und danach Q dazu addiert. Da für beide Skalare in der Binärdarstellung durchschnittlich die Hälfte der Bits Einsen sind, werden im Durchschnitt $2 \cdot 1/2$ ECADD-Operationen durchgeführt. Somit ergibt sich für die Berechnung von $k \cdot P + l \cdot Q$ mit Hilfe der Interleave Methode im Mittel ein Aufwand von

$$n\ ECDBL + n\ ECADD \tag{7}$$

Operationen [Möl01].

Die Interleave Methode lässt sich auch für Skalare in $\mathcal{D}$-Darstellungen anpassen, dazu müssen in einer Vorberechnungsphase alle Punkte $t \cdot P$, $t \in \mathcal{D}_1 \setminus \{0,1\}$ und $t \cdot Q$, $t \in \mathcal{D}_2 \setminus \{0,1\}$ ermittelt werden. Da die ECADD-Operationen getrennt durchgeführt werden, können die Darstellungen für die Skalare bei der Interleave Methode unterschiedlich sein. Zum Beispiel könnte $\mathcal{D}_1 = \{0,1,3\}$ sein und $\mathcal{D}_2 = \{0,1,3,5\}$, in diesem Fall müssten die Punkte $3P$ und $3Q$, $5Q$ vorberechnet werden. Algorithmus 9 gibt die Interleave Methode für eine mehrfache Punktmultiplikation mit zwei Skalaren in einer $\mathcal{D}$-Darstellung an.

Algorithmus 9 Interleave Methode für $\mathcal{D}$-Darstellungen

Benötigt: Skalare $k = (d_{n-1}d_{n-2}\dots d_1d_0)$ mit $d_i \in \mathcal{D}_1$, $l = (d_{n-1}d_{n-2}\dots d_1d_0)$ mit $d_i \in \mathcal{D}_2$ und Punkte P,$Q \in E(\mathbb{F}_p)$.
Liefert: Mehrfache Punktmultiplikation $k \cdot P + l \cdot Q$.

1. {**Vorberechnung**}
2. $S_t \leftarrow t \cdot P$, für alle $t \in \mathcal{D}_1 \setminus \{0\}$
3. $T_t \leftarrow t \cdot Q$, für alle $t \in \mathcal{D}_2 \setminus \{0\}$
4. {**Hauptrechnung**}
5. $R \leftarrow \mathcal{O}$
6. **for** $i = n-1$ down to 0 **do**
7. $\quad R \leftarrow 2 \cdot R$ {ECDBL}
8. $\quad$ **if** $k[i] \neq 0$ **then**
9. $\quad\quad R \leftarrow R + S_{k[i]}$ {ECADD}
10. $\quad$ **end if**
11. $\quad$ **if** $l[i] \neq 0$ **then**
12. $\quad\quad R \leftarrow R + T_{l[i]}$ {ECADD}
13. $\quad$ **end if**
14. **end for**
15. **return** R.

In den Zeilen 9 und 12 ist zu erkennen, dass die Anzahl der ausgeführten ECADD-Operationen in der Hauptrechnung von den Ziffern der Skalardarstellung abhängig ist, die ungleich Null sind. Dadurch ergibt sich für die Evaluation einer Punktmultiplikation $k \cdot P + l \cdot Q$ mit Algorithmus 9 ein durchschnittlicher Aufwand von

$$n\ ECDBL + n \cdot (\mathcal{AHD}(k) + \mathcal{AHD}(l))\ \ ECADD$$

Operationen in der Hauptrechnung. In der Vorberechnungsphase müssen zusätzlich

$$(|\mathcal{D}_1| - 2) + (|\mathcal{D}_2| - 2)$$

Punkte vorberechnet und gespeichert werden, für die weitere Operationen anfallen.

Die Interleave Methode kann natürlich auch für mehrfache Punktmultiplikationen der Form $\sum_{j=1}^{m} k_j \cdot P_j$ modifiziert werden. Demzufolge müssen in Algorithmus 9 nicht nur zwei Skalare bitweise überprüft werden, sondern m Skalare. Damit die einzelnen Algorithmen besser miteinander verglichen werden können, wird davon ausgegangen, dass alle m Skalare in der selben $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ angegeben sind. Infolgedessen beträgt der Aufwand für die Berechnung von $\sum_{j=1}^{m} k_j \cdot P_j$ mit der Interleave Methode im Durchschnitt

$$n\ ECDBL + n \cdot m \cdot \mathcal{AHD}(\mathcal{X})\ \ ECADD$$

Operationen in der Hauptrechnung. Während der Vorberechnung müssen außerdem noch $m \cdot (|\mathcal{D}| - 2)$ Punkte vorberechnet und gespeichert werden. Im Vergleich zur separaten

Berechnung mit Algorithmus 8 mit der durchschnittlichen Aufwandsgleichung (6) werden mit der Interleave Methode einige Operationen eingespart. Die $(m-1)$ ECADD-Operationen werden nicht benötigt, da es keine einzelnen Ergebnisse mehr gibt, die zusammenaddiert werden müssen. Des weiteren reduziert sich die Anzahl der ECDBL-Operationen um den Faktor m [Dah05].

Beispiel 14. *Die folgende Tabelle verbildlicht die Berechnung der mehrfachen Punktmultiplikation* $13 \cdot P + 9 \cdot Q$ *für* $P,Q \in E(\mathbb{F}_p)$ *mit Hilfe der Interleave Methode. Die Skalare* $13 = (301)$ *und* $9 = (1001)$ *sind in einer* $\mathcal{D}$*-Darstellung mit* $\mathcal{D} = \{0,1,3\}$ *gegeben.*

Vorberechnung

$S_1 \leftarrow P$	$T_1 \leftarrow Q$
$S_3 \leftarrow 3P$	$T_3 \leftarrow 3Q$

Hauptrechnung

i	$k[i] = 0301$ $l[i] = 1001$	ECDBL	ECADD	Variable R
				$R = \mathcal{O}$
3	0 1	$2 \cdot \mathcal{O}$	$+T_1$	$R = Q$
2	3 0	$2 \cdot Q$	$+S_3$	$R = 3P + 2Q$
1	0 0	$2 \cdot (3P + 2Q)$		$R = 6P + 4Q$
0	1 1	$2 \cdot (6P + 4Q)$	$+S_1 \quad + T_1$	$R = 13P + 9Q$

In diesem Beispiel werden in der Hauptrechnung genau vier ECDBL- und vier ECADD-Operationen ausgeführt, inklusive der Verdopplung und Addition von $\mathcal{O}$*. Im Vergleich dazu werden bei einer separaten Berechnung mit Algorithmus 8 acht ECDBL- und fünf ECADD-Operationen benötigt, wobei die Anzahl der vorberechneten Punkte bei beiden Vorgehensweisen gleich sind.*

4.3. Shamir Methode

Eine weitere Methode für die mehrfache Punktmultiplikation der Form $k \cdot P + l \cdot Q$ wurde in [Gam85] mit Bezug auf Shamir vorgestellt und ist daher als Shamirs Trick oder die Shamir Methode bekannt. Die Idee dieser Methode ist die ECDBL- und ECADD-Operationen nicht getrennt, sondern für beide Summanden simultan auszuführen. Anschaulich gesehen werden, wie bei der Interleave Methode, die beiden Skalare in ihrer

Binärdarstellung untereinander geschrieben

$$k = d_{n-1}\ d_{n-2}\ \cdots\ d_0$$
$$l = d_{n-1}\ d_{n-2}\ \cdots\ d_0$$

und von links nach rechts bitweise abgearbeitet. Unterschiedliche Bitlängen der Skalare werden mit führenden Nullen in der kürzeren Darstellung ausgeglichen. Für jede Bitspalte wird eine ECDBL-Operation ausgeführt und je nach Bitkombination in der Spalte ein zusätzliches ECADD. Es können dabei vier verschiedene Kombinationen auftreten:

Kombination	1	2	3	4
	0	0	1	1
	0	1	0	1

Für die erste Kombination kann die ECADD-Operation eingespart werden, für die zweite wird der Punkt Q und für die dritte der Punkt P addiert. Wird die letzte Kombination abgearbeitet wird der Punkt $P+Q$ addiert. Damit beim Auftreten der vierten Kombination nicht jedes mal die Summe $P+Q$ neu ermittelt werden muss, wird diese zu Beginn der Methode in der Vorberechnungsphase berechnet und gespeichert. Da nur für drei von vier Kombinationsmöglichkeiten ein ECADD ausgeführt wird, benötigt die Shamir Methode im Durchschnitt

$$n\ ECDBL + n \cdot \frac{3}{4}\ ECADD$$

Operationen zur Berechnung einer mehrfachen Punktmultiplikation $k \cdot P + l \cdot Q$. Zusätzlich muss der Punkt $P+Q$ vorberechnet werden. Im Vergleich zum Berechnungsaufwand (7) mit der Interleave Methode werden demnach durch die Vorberechnung eines zusätzlichen Punktes 1/4 der ECADD Operationen in der Hauptrechnung eingespart [Gam85, MvOV97].

Die Shamir Methode lässt sich auch für Skalare in $\mathcal{D}$-Darstellungen anpassen. Die Anzahl der Punkte, die bei dieser Methode berechnet werden müssen, hängt von der Darstellung der Skalare ab, genauer von den Kombinationsmöglichkeiten der einzelnen Ziffern aus $\mathcal{D}$. Sind die Skalare in der Binärdarstellung mit $\mathcal{D} = \{0,1\}$, muss nur der Punkt $P+Q$ vorberechnet werden. Ist $\mathcal{D} = \{0,1,3\}$ sind schon die sechs Kombinationen $3P$,$3Q$,$P+Q$,$P+3Q$,$3P+Q$,$3P+3Q$ möglich, die vorberechnet werden müssen. Da für die Kombinationen $0P+0Q$,$0P+Q$,$P+0Q$ keine Berechnungen nötig sind, müssen für die Skalare k und l in einer $\mathcal{D}$-Darstellung genau

$$|\mathcal{D}|^2 - 3$$

Punkte vorberechnet werden. Im Gegensatz zur Interleave Methode müssen die Skalare in der selben $\mathcal{D}$-Darstellung gegeben sein, da die Addition simultan für diese ausgeführt wird. Algorithmus 10 zeigt die angepasste Shamir Methode. Für jede Bitspalte, die nur

Algorithmus 10 Shamir Methode für $\mathcal{D}$-Darstellungen

Benötigt: n-Bit lange Skalare k,l in $\mathcal{D}$-Darstellung und Punkte P,$Q \in E(\mathbb{F}_p)$.
Liefert: Mehrfache Punktmultiplikation $k \cdot P + l \cdot Q$.

1. {**Vorberechnung**}
2. $S_{t_1 t_2} \leftarrow t_1 \cdot P + t_2 \cdot Q$, für alle t_1,$t_2 \in \mathcal{D} \setminus \{0\}$
3. $S_{t0} \leftarrow t \cdot P$, für alle $t \in \mathcal{D} \setminus \{0\}$
4. $S_{0t} \leftarrow t \cdot Q$, für alle $t \in \mathcal{D} \setminus \{0\}$
5. {**Hauptrechnung**}
6. $R \leftarrow \mathcal{O}$
7. **for** $i = n-1$ down to 0 **do**
8. $\quad R \leftarrow 2 \cdot R$ {ECDBL}
9. $\quad$ **if** $(k[i], l[i]) \neq (0,0)$ **then**
10. $\quad\quad R \leftarrow R + S_{k[i],l[i]}$ {ECADD}
11. $\quad$ **end if**
12. **end for**
13. **return** R.

aus Nullen besteht, kann in Zeile 10 eine ECADD-Operation eingespart werden, die Anzahl der nicht Null-Spalten gibt das joint Hamming Gewicht ($\mathcal{JHW}$) an. In Relation zur Anzahl der Bitspalten erhält man aus dem ($\mathcal{JHW}$) die joint Hamming Dichte ($\mathcal{JHD}$). Dadurch ergibt sich für die Berechnung von $k \cdot P + l \cdot Q$ mittels Algorithmus 10 ein durchschnittlicher Aufwand von

$$n\ ECDBL + n \cdot \mathcal{AJHD}(k,l)\ ECADD$$

Operationen in der Hauptberechnung.

Auch die Shamir Methode lässt sich für eine mehrfache Punktmultiplikation der Form $\sum_{j=1}^{m} k_j \cdot P_j$ angleichen. Dazu müssen in der Vorberechnunsphase alle Punkte $t_1 P_1 + \ldots + t_m P_m$ mit $t_1, \ldots, t_m \in \mathcal{D}^m$ ermittelt werden, bei denen mindestens zwei der t_j ungleich Null sind. Insgesamt wird die Vorberechnung von

$$|\mathcal{D}|^m - 1 - m$$

Punkten notwendig. Sind die Skalare in einer $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ gegeben, ergibt sich für die Berechnung der Punktmultiplikation $\sum_{j=1}^{m} k_j \cdot P_j$ mittels der Shamir Methode ein durchschnittlicher Aufwand von

$$n\ ECDBL + n \cdot \mathcal{AJHD}_m(\mathcal{X})\ ECADD$$

Operationen in der Hauptberechnungsphase [Dah05].

Beispiel 15. *In der folgenden Tabelle ist eine Berechnung der mehrfachen Punktmultiplikation* $13 \cdot P + 9 \cdot Q$ *für* P,$Q \in E(\mathbb{F}_p)$ *mit Hilfe der Shamir Methode aufgeführt. Die Skalare* $13 = (301)$ *und* $9 = (1001)$ *sind in einer* $\mathcal{D}$*-Darstellung mit* $\mathcal{D} = \{0,1,3\}$ *gegeben.*

Vorberechnung

$S_{1,1} \leftarrow P + Q$	$S_{1,3} \leftarrow P + 3Q$
$S_{3,1} \leftarrow 3P + Q$	$S_{3,3} \leftarrow 3P + 3Q$
$S_{3,0} \leftarrow 3P$	$S_{0,3} \leftarrow 3Q$

Hauptrechnung

i	$k[i] = 0301$ $l[i] = 1001$	ECDBL	ECADD	Variable R
				$R = \mathcal{O}$
3	0 1	$2 \cdot \mathcal{O}$	$+S_{0,1}$	$R = Q$
2	3 0	$2 \cdot Q$	$+S_{3,0}$	$R = 3P + 2Q$
1	0 0	$2 \cdot (3P + 2Q)$		$R = 6P + 4Q$
0	1 1	$2 \cdot (6P + 4Q)$	$+S_{1,1}$	$R = 13P + 9Q$

In diesem Beispiel werden in der Hauptrechnung genau vier ECDBL- und drei ECADD-Operationen ausgeführt, inklusive der Verdopplung und Addition von $\mathcal{O}$. Der deutliche Unterschied zur Interleave Methode wird in der letzten Zeile erkennbar, in der anstatt zwei separater Additionen eine Addition simultan durchgeführt wird. Durch den Mehraufwand in der Vorberechnungsphase wird im Vergleich zu Beispiel 14 genau eine ECADD-Operation in der Hauptberechnung eingespart.

4.4. Vorberechnung von elliptischen Kurvenpunkten

Um Berechnungen einzusparen, werden in den vorgestellten Evaluationsalgorithmen für Multiplikationen von elliptischen Kurvenpunkten mit Skalaren in $\mathcal{D}$-Darstellungen einige Punkte vorberechnet. Die Anzahl der Punkte, die vorberechnet werden muss, lässt sich durch eine geschickt gewählte Ziffernmenge $\mathcal{D}$ deutlich reduzieren.

Im Kapitel 2.2.1 wurde gezeigt, dass die Invertierung eines Kurvenpunktes ohne aufwändige Berechnungen erfolgt. Die Invertierung eines Punktes $P = (x_P, y_P) \in E(\mathbb{F}_p)$ ist lediglich ein Vorzeichenwechsel der y-Koordinate, der inverse Punkt von P ist somit $-P = (x_P, -y_P)$. Wird diese Tatsache in der Vorberechnungsphase ausgenutzt, lassen sich dadurch einige Berechnungen einsparen. Bei einer symmetrischen Ziffernmenge $\mathcal{D} = \{0, \pm 1, \ldots, \pm x\}$ genügt es, wenn nur noch die positiven Punkte vorberechnet werden. Bei der Interleave Methode müssen ursprünglich die Punkte $t_j \cdot P_j$ für alle $t_j \in \mathcal{D} \backslash \{0\}$ und $j = 1, \ldots, m$ vorberechnet werden, wobei die Skalare in der selben $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ gegeben sind. Durch die Symmetrie in der Ziffernmenge ist es

ausreichend, nur die Punkte mit einem positiven $t_j \in \mathcal{D} \setminus \{0,1\}$ zu bestimmen. Wird in der Hauptphase ein Punkt $-t_j \cdot P_j$ benötigt, kann dieser dynamisch (on the fly) invertiert werden. Diese Vorgehensweise reduziert die Punkte, die bei der Interleave Methode in der Vorberechnungsphase ermittelt werden müssen von

$$m \cdot (|\mathcal{D}| - 1) - m \qquad \text{auf} \qquad m \cdot \frac{(|\mathcal{D}| - 1)}{2} - m$$

Punkten, folglich um mehr als die Hälfte [DOT05a, Dah05].

Beispiel 16. *Für die Berechnung von $t_1P_1 + t_2P_2$, $\mathcal{D} = \{0, \pm 1, \pm 3\}$ mittels der Interleave Methode müssten die sechs Punkte $-P_1$,$3P_1$, $-3P_1$, $-P_2$,$3P_2$, $-3P_2$ vorberechnet werden. Da eine dynamische Invertierung fast umsonst ist, reicht es aus, nur die zwei Punkte $3P_1$ und $3P_2$ zu bestimmen.*

In der Shamir Methode müssen alle Punkte $t_1P_1 + \ \ldots + t_mP_m$ mit $t_1, \ \ldots, \ t_m \in \mathcal{D}^m$ vorberechnet werden bei denen mindestens zwei der t_j ungleich Null sind. Wurde zum Beispiel $t_kP_k - t_lP_l$ bereits berechnet, erhält man durch eine Invertierung den Punkt $-t_kP_k + t_lP_l$. Aus diesem Grund ist es möglich alle Kombinationen einzusparen, die durch eine Invertierung bestimmt werden können. Insgesamt werden dadurch die Punkte, die in der Shamir Methode vorberechnet werden müssen, von

$$|\mathcal{D}|^m - 1 - m \qquad \text{auf} \qquad \frac{|\mathcal{D}|^m - 1}{2} - m$$

Punkten reduziert [DOT05a, Dah05].

Beispiel 17. *Soll die Punktmultiplikation $t_1P_1 + t_2P_2$, $\mathcal{D} = \{0, \pm 1\}$ mit Hilfe der Shamir Methode bestimmt werden, müssten die sechs Punkte $-P_1$,$-P_2$,P_1+P_2,$-P_1+P_2$,P_1-P_2,$-P_1-P_2$ vorberechnet werden. Es ist aber ausreichend nur die zwei Punkte P_1+P_2,$-P_1+P_2$ zu bestimmen. Die restlichen vier Punkte können durch eine Invertierung dieser zwei Punkte oder der Punkte P_1,P_2 berechnet werden.*

5. Rekodierung des Skalars

In Kapitel 4 wurde ersichtlich, dass die Anzahl der benötigten ECDBL-Operationen in den Evaluationsalgorithmen genau der Bitlänge n des Skalars entspricht, unabhängig von der $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ des Skalars. Bei einer mehrfachen Punktmultiplikation ist indes die längste Bitlänge der einzelnen Skalare entscheidend. Die Anzahl der benötigten ECADD-Operationen ist je nach Algorithmus von der $\mathcal{AHD}$ oder der $\mathcal{AJHD}$ der Skalare bestimmt und somit sehr wohl von der $\mathcal{D}$-Darstellungsklasse $\mathcal{X}$ abhängig. Wählt man aus diesem Grund für die Skalare eine Darstellung mit einer durchschnittlich niedrigen Hamming-Dichte, lassen sich ECADD-Operationen einsparen und somit die Punktmultiplikation beschleunigen. Wird ein Skalar in eine andere Darstellung überführt, bezeichnet man dies als eine Rekodierung des Skalars (*recoding*). Demnach lässt sich ein Skalarmultiplikationsalgorithmus in drei Phasen aufteilen: in die Vorberechnungsphase, in der die mehrmals benötigten Punkte vorberechnet und gespeichert werden, in die Rekodierungsphase, in welcher der oder die Skalare in eine andere Darstellung umgewandelt werden und in die Evaluationsphase, in der schließlich das Multiplikationsergebnis bestimmt wird. Oftmals werden die Rekodierungsphase und die Evaluationsphase aus Effizienzgründen miteinander kombiniert, die Rekodierung des Skalars erfolgt dabei „on the fly“ während der Evaluationsberechnungen. Die Rekodierungsalgorithmen lassen sich grob in zwei Klassen unterteilen. Die Algorithmen der ersten Klasse rekodieren ein Skalar in eine Darstellung mit möglichst geringer durchschnittlicher Hamming-Dichte ($\mathcal{AHD}$) und sind folglich für eine Beschleunigung der Interleave-Methode geeignet. Die zweite Klasse von Algorithmen rekodiert mehrere Skalare mit dem Ziel möglichst viele Nullspalten und somit eine geringe $\mathcal{AJHD}$ zu erzeugen. Die zweite Klasse von Methoden kann daher zur Beschleunigung der Shamir-Methode genutzt werden [OSSST04].

5.1. Window Recoding

Die meisten Rekodierungsmethoden sind sogenannte window-Methoden, bei denen immer mehrere Bits des Skalars gleichzeitig betrachtet und rekodiert werden. Während der Rekodierung wird der komplette Skalar somit anschaulich durch ein Fenster mit der Breite von ein paar Bits betrachtet. Diese Bits werden rekodiert und das Fenster wandert zu den folgenden Bits des Skalars, die noch nicht rekodiert worden sind. Die Fensterbreite bestimmt dabei die zur Verfügung stehende Ziffernmenge $\mathcal{D}$ für die Rekodierung und dadurch auch die Anzahl der Punkte, die vorberechnet werden müssen.

5.1.1. Fixed-size-sliding-window Recoding Methode

Die fixed-size-sliding-window Methode wird bevorzugt mit der Evaluationsphase kombiniert und findet daher hauptsächlich als „on-the-fly“-Methode Verwendung. Aus diesem

Grund ist sie auch unter den Namen fixed-size-sliding-window exponentiation, 2^w-ary exponentiation oder k-ary exponentiation bekannt [MvOV97, CFA+06]. Bildlich gesehen teilt diese window-Methode den Skalar in der Binärdarstellung von links nach rechts in gleich lange Teilstrings, wobei deren Länge gerade der Fensterbreite w entspricht. Der letzte Teilstring kann natürlich kürzer ausfallen, falls der Binärstring nicht durch w teilbar sein sollte. Jeder Teilstring wird dann separat ausgewertet und entsprechend seinem Wert rekodiert, die Auswertung entspricht dabei der Abbildung $\{0,1\}^w \rightarrow \{0,1,2,\ldots,2^w-1\}$. Der rekodierte Teilstring ist gerade der Wert des Teilstrings mit zusätzlichen führenden Nullen um die Länge beizubehalten. Die Ziffernmenge $\mathcal{D}$ der rekodierten Darstellung ist aus diesem Grund gleich der Bildmenge der Auswertungsabbildung. Für die Fensterbreite w sollte ein Wert größer als eins gewählt werden, da für $w = 1$ die Darstellung unverändert bleibt. Algorithmus 11 beschreibt die Skalarrekodierung mittels der Fixed-size-sliding-window Methode. Die Auswertung eines Teilstrings $(u_i,\ldots,u_j)$ erfolgt mit der Funktion $eval\,(i,j)$ und die Bitlänge des Skalars u sei b. Für die

Algorithmus 11 Fixed-size-sliding-window Recoding Methode

Benötigt: Skalar u und Fensterbreite w.
Liefert: rekodierte Darstellung $\hat{u} = (N_{b-1},\ldots,N_0)$.

1. $i \leftarrow bitlength\,(u) - 1$
2. $t \leftarrow t + 1 \bmod w$
3. **while** $i + 1 \geq w$ **do**
4. $\quad s \leftarrow eval\,(i,i - w + 1)$
5. $\quad N_{i-w+1} \leftarrow s$
6. $\quad i \leftarrow i - w$
7. **end while**
8. **if** $t \neq 0$ **then**
9. $\quad s \leftarrow eval\,(i,0)$
10. $\quad N_0 \leftarrow s$
11. **end if**
12. **return** N.

Fensterbreite $w > 1$ wird die Ziffernauswahl von $\{0,1\}$ um die Ziffern $\{2,3,4,\ldots,2^w-1\}$ erweitert. Für eine Multiplikationsberechnung mit einem durch Algorithmus 11 rekodierten Skalar müssen daher in der Vorberechnungsphase genau $2^w - 2$ Punkte bestimmt werden. Die fixed-size-sliding-window Methode liefert rekodierte Darstellungen mit

$$\frac{1}{w}\left(1 - \frac{1}{2^w}\right)\ \mathcal{AHD}, \tag{8}$$

für $w > 1$ erzeugt selbige Methode also Darstellungen mit einer höheren Dichte an Nullziffern als die Binärdarstellung [MvOV97, Mat07].

Beispiel 18. *Der Skalar*

$$u = 87262 = (101|010|100|110|111|10)_2 \tag{9}$$

soll nun mit dieser Methode und der Fensterbreite $w = 3$ rekodiert werden. In (9) wurde die Binärdarstellung von u bereits von links nach rechts in die Teilstrings zerlegt. Jeder Teilstring wird nun ausgewertet und rekodiert. Der rekodierte Skalar $\hat{u}$ ist somit

$$\hat{u} = \{005|002|004|006|007|02\} = 00500200400600702.$$

Die Ziffernmenge in diesem Beispiel ist $\mathcal{D} = \{0,1,2,3,4,5,6,7\}$, daher müssten für die Evaluationsphase in der Phase der Vorberechnungen die sechs Punkte $\{2P,3P,4P,5P,6P,7P\}$ bestimmt werden. Das Hamming-Gewicht für u in der Binärdarstellung beträgt zehn, wobei die rekodierte Darstellung $\hat{u}$ nur sechs Ziffern aufweist die nicht Null sind.

5.1.2. Sliding-window Recoding Methode

Die sliding-window Methode ist ein weiteres Beispiel für eine Rekodierungsmethode, die vorzugsweise mit der Evaluationsphase kombiniert wird und daher auch als sliding-window exponentiation bekannt ist. Diese Methode verwendet im Gegensatz zur vorangegangenen kein starres Fenster, sondern ein Fenster mit einer dynamischen Breite w. Der Skalar wird hierbei wiederum von links nach rechts in Teilstrings unterteilt, wobei führende und endende Nullziffern übersprungen werden. Infolgedessen entstehen Teilstrings, die kleiner oder gleich der Fensterbreite sind. Durch diese Herangehensweise reduziert sich die mögliche Ziffernmenge $\mathcal{D}$ auf $\{0,1,3,5,\ldots,2^w - 1\}$, da nur noch Teilstrings mit einem ungeraden Wert oder dem Wert Null zustande kommen. Die Anzahl der Punkte, die in der Vorberechnungsphase bestimmt werden müssen, reduziert sich damit auf $2^w - 2/2$ Punkte, also die Hälfte im Vergleich zur fixed-size-sliding-window Methode. Des weiteren liefert die sliding-window Recodingmethode mit einer durchschnittlichen Hamming Dichte von

$$\frac{1}{w+1} \quad \mathcal{AHD} \tag{10}$$

für $w > 1$ bessere Ergebnisse. Die geringere resultierende Dichte an nicht-null Ziffern kann man sich an einem Beispiel klar machen. Mit einer starren Fensterbreite $w = 3$ würden die Teilstrings

$$100|010|100 \qquad \text{in} \qquad 004|002|004|$$

rekodiert werden. Die Methode mit einem dynamischen Fenster der gleichen Breite ändert die Teilstrings jedoch von

$$1|000|101|00| \qquad \text{in} \qquad 1|000|005|00|,$$

weil führende und endende Nullen anders behandelt werden. Algorithmus 12 stellt die Sliding-window Recoding Methode im Detail dar [MvOV97].

Algorithmus 12 Sliding-window Recoding Methode [Mat07]

Benötigt: Skalar u und Fensterbreite w.
Liefert: rekodierte Darstellung $\hat{u} = (N_{b-1}, \ldots, N_0)$.

1. $i \leftarrow bitlength\,(u) - 1$
2. **while** $i \geq 0$ **do**
3. **if** $u_i = 0$ **then**
4. $i \leftarrow i - 1$
5. **else**
6. $j \leftarrow max\,\{i - w + 1, 0\}$
7. **while** $u_j = 0$ **do**
8. $j \leftarrow j + 1$
9. **end while**
10. $s \leftarrow eval\,(i, j)$
11. $N_j \leftarrow s$
12. $i \leftarrow i - w$
13. **end if**
14. **end while**
15. **return** N.

Beispiel 19. *Der Skalar*

$$u = 87262 = (101|0|101|00|11|0|111|1|0)_2 \tag{11}$$

soll nun mit Algorithmus 12 und einer Fensterbreite $w = 3$ rekodiert werden. In (11) erkennt man deutlich die zu rekodierenden Teilstrings und die Nullstrings, die übersprungen werden. Der rekodierte Skalar $\hat{u}$ ist somit

$$\hat{u} = \{005|0|005|00|03|0|007|1|0\} = 00500050003000710.$$

Die Ziffernmenge in diesem Beispiel ist $\mathcal{D} = \{0,1,3,5,7\}$, daher müssten für die Evaluationsphase in der Phase der Vorberechnungen nur noch die drei Punkte $\{3P, 5P, 7P\}$ bestimmt werden. Das Hamming-Gewicht für $\hat{u}$ beträgt fünf, wohingegen die rekodierte Darstellung von u mittels der fixed-size-sliding-window Methode ein Gewicht von sechs aufweist.

5.1.3. Fractional-window Recoding Methode

Wurde ein Skalar mittels der sliding-window Methode rekodiert, müssen $2^{w-1} - 1$ Punkte vorberechnet und für die Evaluationsphase gespeichert werden. Die Anzahl der zu speichernden Punkte wächst für zwei aufeinander folgende Fensterbreiten w und $w + 1$ exponentiell. Bei Systemen mit beschränktem Speicherplatz kann es folglich vorkommen, dass die geringere Fensterbreite gewählt werden muss, obwohl noch Speicherplatz

vorhanden wäre, welcher aber nicht für die nächsthöhere Breite ausreicht und somit vergeudet wird. Aus diesem Grund wurde in [Möl04] die fractional-window Methode vorgeschlagen, bei der die Anzahl der zu speichernden Punkte frei wählbar ist und der vorhandene Speicherplatz optimal genutzt werden kann. Diese Rekodierungsmethode verwendet vorzeichenbehaftete Darstellungen (*signed digit representation*) mit der Ziffernmenge $\mathcal{D}_m = \{0, \pm 1, \pm 3, \ldots, \pm m\}$, wobei $m \geq 1$ und ungerade ist. Für ein $m \in \{0{,}1{,}3, \ldots, 2^x - 1\}$ ergibt sich für die fractional-window Methode eine Fensterbreite von

$$\omega_m = \lfloor \log_2 m \rfloor + 1.$$

Für alle anderen m ergibt sich ein Teilfenster (*fraction window*) zwischen den zwei aufeinander folgenden Fensterbreiten ω_m und

$$W_m = \omega_m + 1.$$

Die hier vorgestellte fractional-window Methode bearbeitet den Skalar von links nach rechts und greift bei der Rekodierung auf die Eigenschaften der MOF-Darstellung zurück, die im Kapitel 5.3 genauer dargestellt werden. Die Grundidee der Rekodierung mit einem fractional-window ist, den Skalar in eine MOF-Darstellung zu überführen und von links nach rechts mittels der sliding-window Methode zu rekodieren. Dabei wird zuerst versucht, den Wert des Fensters mit der Breite W_m durch eine Ziffer aus $\mathcal{D}_m$ zu ersetzen. Sollte der passende Wert nicht enthalten sein, wird die Fensterbreite auf ω_m verringert. Die MOF-Eigenschaften stellen sicher, dass bei der Breite ω_m der absolute Wert höchstens 2^{ω_m} oder $2^{\omega_m} - 1$ beträgt. Da alle ungeraden Ziffern bis zur unteren Fensterbreite ω_m in $\mathcal{D}_m$ enthalten sind, existiert für jeden ungeraden Wert in diesem Fenster eine passende Ziffer. Bei einem geraden Wert wird einfach solange durch zwei geteilt, bis dieser ungerade ist. Ein Beispiel mit $u = 22369$ soll die Vorgehensweise verdeutlichen. Sei die Ziffernmenge $\mathcal{D}_5 = \{0, \pm 1, \pm 3, \pm 5\}$, für $m = 5$ folgt $\omega_5 = 3$ und $W_5 = 4$. Die MOF-Darstellung erhält man leicht durch die binäre Rechnung $2 \cdot (u)_2 \ominus (u)_2$, wobei $\ominus$ die Bitweise Subtraktion darstellt. Für u erhält man in diesem Fall die MOF-Darstellung $1\bar{1}1\bar{1}100\bar{1}10\bar{1}0001\bar{1}$. Die zulässigen Fensterbreiten ergeben die Konstellation

$$\boxed{1\bar{1}1\bar{1}}\ \boxed{100}\ \boxed{\bar{1}10\bar{1}}\ 000\ \boxed{1\bar{1}}.$$

Das erste Fenster ganz links kann vier Bits breit sein, da dessen Wert einer Ziffer in $\mathcal{D}_5$ entspricht. Beim zweiten Fenster ist die gleiche Bitbreite nicht möglich, da die Ziffer sieben in der Ziffernmenge nicht enthalten ist, daher wird das Fenster auf ω_5 verringert. Weil aber nur ungerade Ziffern existieren, bleibt dieser Teil unverändert. Beim darauf folgenden Fenster ist wiederum die Breite W_5 möglich, die folgenden Nullen werden übersprungen und das letzte Fenster ist nur zwei Bits breit. Die vollständig rekodierte Darstellung sieht dann folgendermaßen aus:

$$\boxed{5}\ \boxed{100}\ \boxed{000\bar{5}}\ 000\ \boxed{01}.$$

Die fractional-window Methode ist im Detail in Algorithmus 13 aufgelistet. Das i-te Bit der MOF-Darstellung erhält man durch die bitweise Subtraktion $b_i = u_{i-1} - u_i$. In den Zeilen 8 und 11 wird die Tatsache ausgenutzt, dass für die Bits b_i mit $i \geq h$ der MOF-Darstellung folgendes gilt:

$$\begin{aligned}(b_i \dots b_{h-1})_2 &= (u_{i-1} \dots u_{h-1})_2 - (u_i \dots u_h)_2 \\ &= (u_{i-1} \dots u_h)_2 \cdot 2 + u_{h-1} - u_i \cdot 2^{i-h} - (u_{i-1} \dots u_h)_2 \\ &= -u_i \cdot 2^{i-h} + (u_{i-1} \dots u_h)_2 + u_{h-1} \\ &= (\overline{u}_i u_{i-1} \dots u_h)_2 + u_{h-1}.\end{aligned}$$

Algorithmus 13 Fractional-window Recoding Methode

Benötigt: Skalar u und größte positive Ziffer m aus $\mathcal{D}_m$.
Liefert: rekodierte Darstellung $\hat{u} = (b_l, \dots, b_0)$.

1. $i \leftarrow bitlength\,(u) + 1$
2. $l \leftarrow 0$
3. **while** $i \geq 0$ **do**
4. **if** $u_i = u_{i-1}$ **then** {i-te Ziffer in Mof-Darstellung ist Null}
5. $i \leftarrow i - 1$
6. **else** {i-te Ziffer in Mof-Darstellung ist ungleich Null}
7. $W \leftarrow W_m$
8. $d \leftarrow (\overline{u}_i u_{i-1} \dots u_{i-W+1})_2 + u_{i-W}$
9. **if** $d \bmod 2 = 1 \wedge |d| > m$ **then** {d ungerade und nicht in $\mathcal{D}_m$ enthalten}
10. $W \leftarrow \omega_m$
11. $d \leftarrow (\overline{u}_i u_{i-1} \dots u_{i-W+1})_2 + u_{i-W}$
12. **end if**
13. $next\_i \leftarrow i - W$
14. $i \leftarrow next\_i + 1$
15. **while** $d \bmod 2 = 0$ **do** {d gerade}
16. $i \leftarrow i + 1$
17. $d \leftarrow d/2$
18. **end while**
19. $b_i \leftarrow d$ {d ungerade und in $\mathcal{D}_m$ enthalten}
20. **if** $i > l$ **then**
21. $l \leftarrow i$
22. **end if**
23. $i \leftarrow next\_i$
24. **end if**
25. **end while**
26. **return** $(b_l, \dots, b_0)$.

Zudem gilt im Algorithmus $u_i = 0$ für alle Bits des Skalars mit $i < 0$. Die fractional-window Methode erzeugt für ein ungerades $m \geq 1$ eine vorzeichenbehaftete Darstellung

mit der Ziffernmenge $\mathcal{D}_m = \{0, \pm 1, \pm 3, \ldots, \pm m\}$ und einer durchschnittlichen Hamming Dichte von

$$\frac{1}{W_m + \frac{1+m}{2^{\omega_m - 1}} - 1} \quad \mathcal{AHD}.$$

Für ein m der Form $m = 2^x - 1$ mit $x \geq 1$ erhält man eine Darstellung mit

$$\frac{1}{W_m + 1} \quad \mathcal{AHD},$$

welche somit die gleiche durchschnittliche Hamming-Dichte liefert, wie eine entsprechende W_m-NAF oder W_m-MOF Darstellung. Einige Beispielwerte für die AHD dieser Methode sind in Tabelle 3 aufgelistet.

Der Autor hat in [Möl04] die Minimalität der fractional-window Rekodierungsmethode bewiesen. Somit existiert keine andere Darstellung einer Ganzzahl mit Ziffern aus $\mathcal{D}_m$ mit einem geringeren Hamming-Gewicht, als die mit der fractional-window Methode erhaltene Darstellung. Der Beweis schließt die Minimalität der AHD dieser Darstellung mit ein. Kommen mittels der fractional-window Methode rekodierte Skalare in Evaluationsalgorithmen zu Anwendung, muss nur noch die Punktemenge $\{3P, 5P, \ldots, mP\}$ vorberechnet werden. Die Gesamtzahl der Punkte, die vorberechnet und gespeichert werden muss beläuft sich daher auf

$$\frac{m-1}{2}$$

Punkte. Somit wächst die Anzahl der Punkte nicht mehr exponentiell und lässt sich exakt dem zur Verfügung stehenden Speicherplatz anpassen [Möl04].

m	ω_m	W_m	$\mathcal{AHD}$	
1	1	2	1/3	$\approx 0{,}3333$
2	2	3	2/7	$\approx 0{,}2857$
3	2	3	1/4	$= 0{,}25$
4	3	4	4/17	$\approx 0{,}2353$
5	3	4	2/9	$\approx 0{,}2222$
6	3	4	4/19	$\approx 0{,}2105$
7	3	4	1/5	$= 0{,}2$
8	4	5	8/41	$\approx 0{,}1951$
9	4	5	4/21	$\approx 0{,}1905$
10	4	5	8/43	$\approx 0{,}1860$
11	4	5	2/11	$\approx 0{,}1818$

Tabelle 3: Beispielwerte für die $\mathcal{AHD}$ der Fractional-window Rekodierungsmethode

5.2. *w*NAF Darstellung

Die wohl bekannteste vorzeichenbehaftete Darstellung ist die non-adjacent-form (NAF) mit der Ziffernmenge $\mathcal{D} = \{-1,0,1\}$, bei der höchstens eine von zwei aufeinander folgenden Ziffern nicht Null ist. Oder anders ausgedrückt, in dieser Darstellung kommen keine zwei benachbarten nicht-Null Ziffern vor (non-adjacent). In [Rei60] hat Reitwiesner bewiesen, dass für jede Ganzzahl eine eindeutige NAF-Darstellung mit 1/3 $\mathcal{AHD}$ existiert. In [MO90] schlugen Morain und Olivos vor, die skalare Punktmultiplikation bei elliptischen Kurvenpunkten durch die Anwendung von NAF zu beschleunigen. In [WMPW98] schlugen die Autoren vor, ein Skalar in einer NAF-Darstellung nochmals mit der sliding-window Methode zu rekodieren, um eine Darstellung mit noch geringerer Hamming Dichte zu erhalten. Einen anderen Ansatz liefert Solinas in [Sol00], indem er die NAF-Eigenschaften für w benachbarte Bits erweitert und ein verallgemeinertes NAF-Rekodierungsverfahren vorstellt. Bei diesem wird nicht im nachhinein eine Fenstermethode auf eine non-adjacent-form angewandt, sondern direkt aus dem Skalar eine verallgemeinerte wNAF-Darstellung gebildet.

Definition (wNAF)**.** *Eine Folge von vorzeichenbehafteten Ziffern wird als wNAF bezeichnet, falls alle drei folgenden Bedingungen erfüllt sind:*

1. *Die höchste Ziffer, die ungleich Null ist, hat ein positives Vorzeichen.*
2. *Unter allen w aufeinander folgenden Ziffern ist höchstens eine Ziffer ungleich Null.*
3. *Jede Ziffer, die ungleich Null ist, ist ungerade und größer als -2^{w-1}, aber kleiner als 2^{w-1}.*

Algorithmus 14 beschreibt das Rekodierungsverfahren aus [Sol00] mit dem man eine wNAF-Darstellung erhält. Im Algorithmus bezeichnet *mods* eine modulo-Funktion, die auch negative Werte liefert, a *mods* b ist dabei folgendermaßen definiert:

$$a \bmod b \qquad \text{und} \qquad -\frac{b}{2} \leq a < \frac{b}{2}.$$

Das Ergebnis von *mods* liefert somit den Rest mit dem kleinsten Absolutwert. Die Ziffernmenge einer wNAF wird durch die Fensterbreite w mit

$$\mathcal{D}_w = \left\{0, \pm 1, \pm 3, \ldots, \pm 2^{w-1} - 1\right\}$$

bestimmt. Muir und Stinson haben in [MS06] bewiesen, dass für jede Ganzzahl eine eindeutige wNAF-Darstellung existiert, die höchstens ein Bit länger als die entsprechende Binärdarstellung ist. Für eine Fensterbreite von $w = 2$ erhält man gerade die ursprüngliche NAF-Darstellung, daher gilt 2NAF = NAF. Die durchschnittliche Hamming-Dichte der non-adjacent-form mit Fensterbreite w lässt sich folgendermaßen bestimmen:

$$\mathcal{AHD}\left(wNAF\right) = \frac{1}{w+1}.$$

Algorithmus 14 wNAF Recoding Methode

Benötigt: Skalar u und Fensterbreite w.
Liefert: rekodierte Darstellung $\hat{u} = (b_l, \ldots, b_0)$.

1. $i \leftarrow 0$
2. **while** $u \geq 1$ **do**
3. **if** $u \bmod 2 = 0$ **then**
4. $b_i \leftarrow 0$
5. **else**
6. $b_i \leftarrow u \; mods \; 2^w$
7. $u \leftarrow u - b_i$
8. **end if**
9. $u \leftarrow u/2$
10. $i \leftarrow i + 1$
11. **end while**
12. **return** $(b_l, \ldots, b_0)$.

In [MS06] haben die Autoren bewiesen, dass das Hamming Gewicht der wNAF-Darstellung für jedes w minimal ist. Damit wurde bewiesen, dass keine andere Darstellung einer Ganzzahl mit Ziffern aus $\mathcal{D}_w$ existiert, die weniger nicht-Null-Ziffern aufweisen kann, als deren wNAF. Dieser Beweis schließt die Minimalität der $\mathcal{AHD}$ der wNAF mit ein.

Ein Vergleich mit der sliding-window Methode zeigt, dass die durchschnittliche Hamming-Dichte bei der wNAF-Darstellung unverändert bleibt. Der große Vorteil der wNAF ist, wie schon in Kapitel 4.4 beschrieben, die vorzeichenbehaftete Ziffernmenge, durch die sich der Vorberechnungsaufwand deutlich reduziert. Der Nachteil der w-non-adjacent-form ist die Rekodierungsrichtung, welche vom niedrigsten zum höchstwertigen Bit verläuft und somit von rechts nach links. Aus Effizienzgründen werden vorzugsweise Evaluationsalgorithmen eingesetzt, die den Skalar von links nach rechts abarbeiten. Verläuft die Rekodierung in entgegengesetzter Richtung, muss der Skalar vor der Evaluationsphase vollständig rekodiert werden. Arbeitet die Rekodierungsphase aber in der gleichen Richtung, können Rekodierung und Evaluation miteinander kombiniert werden und die Zwischenspeicherung des vollständig rekodierten Skalar entfällt. Dadurch lässt sich bei einem n-Bit langem Skalar Arbeitsspeicher in der Größenordnung von $\mathcal{O}(n)$ einsparen [OSSST04].

Beispiel 20. *Abschließend soll nun die Ganzzahl $u = 87262$ für verschiedene Fensterbreiten in die entsprechende wNAF überführt werden.*

	$u =$	10101010011011110
$w = 2$	$NAF(u) =$	$101010101001\bar{0}001\bar{0}10$
$w = 3$	$3NAF(u) =$	$100\bar{3}00300\bar{3}001\bar{0}001\bar{0}10$
$w = 4$	$4NAF(u) =$	$500050000700\bar{1}0$

5.3. MOF Darstellung

In [OSSST04] stellen die Autoren die mutual-opposite-form (MOF), eine vorzeichenbehaftete Binärdarstellung mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$, vor. Die Anwendung der sliding-window Methode auf diese Darstellungsform führt zur einer wMOF, die exakt die gleiche Hamming-Dichte liefert wie die wNAF. Im Gegensatz zur dieser kann die wMOF-Darstellung aber vom höchstwertigen Bit aus, also von links nach rechts, generiert werden. Diese Darstellung erlaubt somit eine dynamische (on-the-fly) Rekodierung bei den Evaluationsalgorithmen, die den Skalar von links nach rechts abarbeiten.

In der folgenden Definition einer MOF werden zwei Ziffern, die nicht Null sind als benachbart betrachtet, auch wenn diese von Nullen getrennt sind.

Definition (MOF)**.** *Eine Folge von vorzeichenbehafteten Ziffern aus der Menge* $\mathcal{D} = \{0, \pm 1\}$ *wird genau dann als mutual-opposite-form (MOF) bezeichnet, wenn die folgenden Bedingungen erfüllt sind:*

1. *Die Vorzeichen von zwei benachbarten nicht-Null Ziffern (ohne Berücksichtigung von Nullen), sind verschieden.*

2. *Die höchstwertige nicht-Null Ziffer ist* 1 *und die niedrigste nicht-Null Ziffer ist* $\bar{1}$*, es sei denn alle Ziffern sind Null.*

In einer MOF haben nach dieser Definition die Ziffern ungleich Null beidseitig benachbart entgegengesetzte Vorzeichen (mutual-opposite). Die Autoren haben in [OSSST04] bewiesen, dass für jede Ganzzahl eine eindeutige MOF-Darstellung existiert, die höchstens ein Bit länger als die entsprechende Binärdarstellung ist. Ferner wurde gezeigt, dass die durchschnittliche Hamming-Dichte von MOF, die der Binärdarstellung entspricht:

$$\mathcal{AHD}\,(MOF) = \frac{1}{2}$$

Eine MOF für eine Ganzzahl lässt sich recht leicht aus dessen Binärdarstellung generieren. Sei d der binäre n-Bit lange String, durch die Berechnung $\mu = 2 \cdot d \ominus d$ wird die dazugehörige MOF-Darstellung μ erzeugt, wobei $\ominus$ die bitweise Subtraktion darstellt. Die i-te Ziffer der MOF erhält man demzufolge durch $\mu_i = d_{i-1} - d_i$ für $i = 1, \ldots, n-1$ und $\mu_n = d_{n-1}$, $\mu_0 = -d_0$. Folgende Tabelle soll die Vorgehensweise verdeutlichen:

$$\begin{array}{rcccccccccccc}
 & 2d & = & d_{n-1} & | & d_{n-2} & | \ldots | & d_{i-1} & | \ldots | & d_1 & | & d_0 & | & \\
\ominus & d & = & & | & d_{n-1} & | \ldots | & d_i & | \ldots | & d_2 & | & d_1 & | & d_0 \\
\hline
 & \mu & = & d_{n-1} & | & d_{n-2} - d_{n-1} & | \ldots | & d_{i-1} - d_i & | \ldots | & d_1 - d_2 & | & d_0 - d_1 & | & -d_0
\end{array}$$

Zur Berechnung des i-ten Bits von μ_i werden nur zwei aufeinander folgende Bits des Binärstrings benötigt, somit kann jede MOF-Ziffer unabhängig von den anderen ermittelt

werden. Die Berechnungen können dabei am höchstwertigen oder auch am niedrigsten Bit beginnen. Algorithmus 15 beschreibt die Variante von links nach rechts [OSSST04].

Algorithmus 15 MOF Recoding Methode

Benötigt: Skalar $u = (d_{n-1}d_{n-2}\dots d_1d_0)_2$.
Liefert: rekodierte MOF-Darstellung $\mu = (\mu_n, \dots, \mu_0)$ von u.

1. $\mu_n \leftarrow d_{n-1}$
2. **for** $i = n-1$ down to 1 **do**
3. $\quad \mu_i \leftarrow d_{i-1} - d_i$
4. **end for**
5. $\mu_0 = -d_0$
6. **return** $(\mu_n, \mu_{n-1}, \dots, \mu_1, \mu_0)$.

Beispiel 21. *Für die Ganzzahl $u = 87262$ wir die zugehörige MOF-Darstellung mittels Algorithmus 15 anschaulich folgendermaßen bestimmt:*

$$\begin{array}{rcl} 2d & = & 1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0 \\ \ominus\ d & = & \quad 1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0 \\ \hline \mu & = & 1\,\bar{1}\,1\,\bar{1}\,1\,\bar{1}\,1\,\bar{1}\,0\,1\,0\,\bar{1}\,1\,0\,0\,0\,\bar{1}\,0 \end{array}$$

Demzufolge ist der vorzeichenbehaftete Binärstring $\mu = 1\bar{1}1\bar{1}1\bar{1}1\bar{1}010\bar{1}1000\bar{1}0$ die MOF von $u = 87262$.

5.3.1. *w*MOF Darstellung

Mit Hilfe der sliding-window Methode mit Fensterbreite $w \geq 2$ lässt sich aus einer MOF eine wMOF-Darstellung erzeugen, die wie folgt definiert ist:

Definition (wMOF)**.** *Eine Folge von vorzeichenbehafteten Ziffern wird genau dann als width w mutual-opposite-form (wMOF) bezeichnet, wenn die folgenden Bedingungen erfüllt sind:*

1. *Die höchstwertige Ziffer, die nicht Null ist, ist positiv.*

2. *Alle nicht-Null Ziffern x, außer die Ziffer mit dem niedrigsten Stellenwert, sind von w-1 Nullen folgendermaßen umgeben:*

   - *falls $2^{k-1} < |x| < 2^k$ für ein k mit $2 \leq k \leq w-1$ gilt, stehen k Nullen vor und $w-k-1$ Nullen nach der Ziffer x.*

$$\underbrace{0\ldots0}_{k}\,x\,\underbrace{0\ldots0}_{w-k-1}$$

- *falls* $|x| = 1$ *gilt, sind zwei Konstellationen möglich:*
  - *Entweder folgen dem* x $w-1$ *Nullen.*

    $$x\,\underbrace{0\ldots0}_{w-1}$$

    Die nächstniedrigere nicht-Null Ziffer hat dann ein umgekehrtes Vorzeichen zur Ziffer x*.*
  - *Oder vor dem* x *steht eine Null und* $w-2$ *Nullen dahinter.*

    $$0x\,\underbrace{0\ldots0}_{w-2}$$

    Die nächstniedrigere nicht-Null Ziffer hat dann das gleiche Vorzeichen wie die Ziffer x*.*

  Falls x *die niedrigste nicht-Null Ziffer ist, kann die Anzahl der angrenzenden Nullen auf der rechten Seite von* x *kleiner sein als oben angegeben. Es ist nicht möglich, dass die letzte nicht-Null Ziffer eine* 1 *ist, während sie einer nicht-Null Ziffer nachfolgt.*

3. *Jede Ziffer ungleich Null ist ungerade und größer als* -2^{w-1} *aber kleiner als* 2^{w-1}*.*

Mit der Ausnahme für die Ziffer mit dem niedrigsten Stellenwert wird der Fall berücksichtigt, in dem das letzte Fenster kleiner als die Breite w ist und somit nicht genau $w-1$ Nullen die letzte Ziffer umgeben können. Im letzten rekodierten Fenster kann die letzte nicht-Null Ziffer nur eine 1 sein, wenn das Fenster in das Bitmuster $(010\ldots0)$ rekodiert wurde. Eine Rekodierung in das Bitmuster $(10\ldots0)$ ist im letzten Fenster nicht möglich, da die letzte nicht-Null Ziffer einer MOF immer eine Eins mit negativem Vorzeichen ist. Aus diesem Grund ist es ist nicht möglich, dass in einer wMOF die letzte nicht-Null Ziffer eine 1 ist, während sie einer nicht-Null Ziffer nachfolgt. Gemäß der Definition verwendet eine wMOF die Ziffernmenge

$$\mathcal{D}_w = \left\{0, \pm 1, \pm 3, \ldots, \pm 2^{w-1} - 1\right\},$$

demzufolge genau die gleichen Ziffern wie die wNAF-Darstellung [OSSST04].

Algorithmus 16 beschreibt die Vorgehensweise, mit der man eine wMOF für einen positiven Skalar u erhält. Um einen negativen Skalar $-u$ in dessen wMOF-Darstellung zu

Algorithmus 16 wMOF Recoding Methode

Benötigt: Skalar $u = (d_{n-1}d_{n-2}\dots d_1d_0)_2$ und Fensterbreite w.
Liefert: rekodierte wMOF-Darstellung $\mu = (\mu_n, \dots, \mu_0)$ von u.

1. $d_{-1} \leftarrow 0$
2. $d_n \leftarrow 0$
3. $i \leftarrow n$
4. **while** $i \geq w - 1$ **do**
5. **if** $d_i = d_{i-1}$ **then** {MOF-Fenster beginnt mit einer Null}
6. $\mu_i \leftarrow 0$
7. $i \leftarrow i - 1$
8. **else** {MOF-Fenster beginnt mit einer Ziffer ungleich Null}
9. $(\mu_i,\mu_{i-1},\dots,\mu_{i-w+1}) \leftarrow \text{Tabelle}_w\,(d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{i-w} - d_{i-w+1})$
10. $i \leftarrow i - w$
11. **end if**
12. **end while**
13. **if** $i \geq 0$ **then** {letztes Fenster ist kleiner als w}
14. $(\mu_i,\mu_{i-1},\dots,\mu_0) \leftarrow \text{Tabelle}_{i+1}\,(d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_0 - d_1, -d_0)$
15. **end if**
16. **return** $(\mu_n,\mu_{n-1},\dots,\mu_1,\mu_0)$.

überführen, wird der positive Wert u rekodiert und diese im Anschluss invertiert. Die wMOF recoding Methode arbeitet den Skalar u von links nach rechts bitweise ab, berechnet die MOF-Darstellung und rekodiert aus dieser jeweils ein MOF-Fenster der Breite w. Führende Nullen in einem MOF-Fenster werden dabei übersprungen, somit beginnt jedes Fenster mit einer Ziffer ungleich Null. Wie bei der sliding window methode werden auch endende Nullen im Fenster übersprungen, so dass nur ungerade Ziffern für die Rekodierung nötig sind. Die w MOF-Ziffern $(m_i,m_{i-1},\dots,m_{i-w+1})$ für ein MOF-Fenster erhält man durch die bitweise Subtraktion $(d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{i-w} - d_{i-w+1})$, der w aufeinander folgenden Ziffern des Skalars u. Sei m_l die niedrigste Ziffer ungleich Null in diesem Fenster, so dass $m_l \neq 0$ und $m_p = 0$ für $p = l - 1, \dots, i - w + 1$ gilt. In diesem Fall wird die Ziffer m_l durch den Dezimalwert des MOF-Fensters ersetzt und die restlichen $w - 1$ Ziffern in diesem Fenster auf Null gesetzt. Diese Umformung ist immer möglich, ohne die Ziffernmenge $\mathcal{D}_w$ zu überschreiten, da der Dezimalwert von w aufeinander folgenden MOF-Ziffern höchstens $2^{w-1} - 1$ sein kann, was der MOF $(1,0,\dots,0,\bar{1})$ entspricht. Algorithmus 16 berechnet die Dezimalwerte nicht für jedes Fenster separat, sondern erstellt zu Beginn eine Hilfstabelle $Tabelle_w$ mit allen möglichen Umformungen der MOF-Fenster mit Breite w. Da jedes MOF-Fenster mit einer Ziffer ungleich Null beginnt, müssen in der Tabelle nur Kombinationen gespeichert werden, deren höchstwertige Bits nicht Null sind. Zur Verdeutlichung folgt ein Beispiel für die Umformungstabelle

T_4:

$$1000 \mapsto 1000 \quad 1\bar{1}00 \mapsto 0100 \quad \left.\begin{matrix}1\bar{1}10\\10\bar{1}0\end{matrix}\right\} \mapsto 0030 \quad \left.\begin{matrix}1\bar{1}01\\1\bar{1}1\bar{1}\end{matrix}\right\} \mapsto 0005 \quad \left.\begin{matrix}100\bar{1}\\10\bar{1}1\end{matrix}\right\} \mapsto 0007$$

$$\bar{1}000 \mapsto \bar{1}000 \quad \bar{1}100 \mapsto 0\bar{1}00 \quad \left.\begin{matrix}\bar{1}1\bar{1}0\\\bar{1}010\end{matrix}\right\} \mapsto 00\bar{3}0 \quad \left.\begin{matrix}\bar{1}10\bar{1}\\\bar{1}1\bar{1}1\end{matrix}\right\} \mapsto 00\bar{5}0 \quad \left.\begin{matrix}\bar{1}001\\\bar{1}01\bar{1}\end{matrix}\right\} \mapsto 000\bar{7}$$

Für die Fensterbreite $w = 4$ ist die Tabelle T_4 aufgrund der MOF-Eigenschaften vollständig. Ein Vergleich der Umformungstabelle T_4 mit der wMOF-Definition, verdeutlicht die aufgelisteten Eigenschaften unter Punkt zwei. In der letzten if-Schleife in Algorithmus 16 wird der Fall berücksichtigt, dass das letzte MOF-Fenster kleiner als w sein kann. Damit auch dieses Fenster korrekt rekodiert werden kann, werden zusätzlich die Umformungstabellen T_{w-1},T_{w-2},...,T_2 benötigt. Effiziente Methoden zur Berechnung und Speicherung der benötigten Umformungstabellen werden in [OSSST04] vorgestellt.

Die Autoren haben in [OSSST04] bewiesen, dass sich für die wMOF eine durchschnittliche Hamming-Dichte von

$$\mathcal{AHD}\left(wMOF\right) = \frac{1}{w+1}$$

ergibt. Des Weiteren wurde nachgewiesen, dass für jedes Skalar eine eindeutige wMOF-Darstellung existiert und diese höchstens ein Bit länger ist als die entsprechende Binärdarstellung. In [Ava04] wurde bewiesen, dass das Hamming Gewicht der wMOF-Darstellung für jedes w minimal ist. Damit wurde belegt, dass keine andere Darstellung einer Ganzzahl mit Ziffern aus $\mathcal{D}_w$ existiert, die weniger nicht-Null-Ziffern aufweisen kann als deren wMOF. Dieser Beweis schließt die Minimalität der $\mathcal{AHD}$ der wMOF mit ein.

Beispiel 22. *Die Ganzzahl $u = 87262$ wird nun mit Algorithmus 16 und der Fensterbreite $w = 3$ in die 3MOF-Darstellung überführt. Für u erhält man*

$$MOF(u) = \left(1\bar{1}1|\bar{1}1\bar{1}|1\bar{1}0|10\bar{1}|100|0|\bar{1}0\right),$$

wobei die resultierenden MOF-Fenster schon markiert sind. Die Umformung der einzelnen MOF-Fenster ergibt dann die folgende Darstellung:

$$3MOF(u) = \left(003|00\bar{3}|010|003|100|0|\bar{1}0\right).$$

Abschließend soll u für verschiedene Fensterbreiten in die entsprechende wMOF überführt werden.

	$MOF\left(u\right) =$	$1\bar{1}1\bar{1}1\bar{1}1\bar{1}010\bar{1}1000\bar{1}0$
$w = 2$	$2MOF\left(u\right) =$	$0101010101001\bar{0}00\bar{1}0$
$w = 3$	$3MOF\left(u\right) =$	$00300\bar{3}01000310\bar{0}0\bar{1}0$
$w = 4$	$4MOF\left(u\right) =$	$000500050000700\bar{1}0$

5.4. Joint Sparse Form

Wie in Kapitel 4.3 gezeigt wurde, lassen sich für jede Nullspalte bei der Evaluation einer mehrfachen Punktmultiplikation mittels der Shamir Methode Berechnungen einsparen. Aus diesem Grund sind Darstellungen mit einer geringen durchschnittlichen joint Hamming-Dichte für diese Methode von Vorteil. Dieses Kapitel beschäftigt sich mit Rekodierungsverfahren, die eine geringe $\mathcal{AJHD}$ erzeugen.

5.4.1. Joint Sparse Form für zwei Ganzzahlen

Die durchschnittliche Hamming-Dichte für die Binärdarstellung beträgt $1/2$, weil die Ziffern Null und Eins gleich wahrscheinlich auftreten. Für zwei untereinander geschriebene Ganzzahlen in der Binärdarstellung ist daher die durchschnittliche Wahrscheinlichkeit für Nullspalten genau $1/2 \cdot 1/2 = 1/4$. Die $\mathcal{AJHD}$ für zwei Ganzzahlen in der Binärdarstellung ergibt demzufolge $\mathcal{AJHD}_2(\text{binär}) = 1 - 1/4 = 3/4$.

Die NAF-Darstellung liefert mit $1/3$ eine wesentlich günstigere durchschnittliche Hamming-Dichte, die Wahrscheinlichkeit für eine Null ist hierbei mit $2/3$ gegeben. Sind die zwei untereinander geschriebenen Ganzzahlen in der NAF-Darstellung, ergibt sich im Durchschnitt eine Wahrscheinlichkeit für Nullspalten von $2/3 \cdot 2/3 = 4/9$. Folglich ist die durchschnittliche joint Hamming-Dichte für zwei Zahlen in der NAF $\mathcal{AJHD}_2(\text{NAF}) = 1 - 4/9 = 5/9$, was eine deutliche Verbesserung gegenüber der Binärdarstellung ausmacht.

Solinas hat in [Sol01] die Joint Sparse Form (JSF) vorgestellt, mit dem dieser Wert noch verbessert werden kann, da die nicht-Null-Spalten in dieser Form dünn besetzt (sparse) sind.

Definition (Joint Sparse Form)**.** *Seien r_0,r_1 Ganzzahlen und n die maximale Bitlänge der beiden Zahlen, dann ist die Matrix*

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} d_{0,n} & \dots & d_{0,0} \\ d_{1,n} & \dots & d_{1,0} \end{pmatrix}$$

genau dann in einer Joint Sparse Form, wenn die folgenden Bedingungen alle erfüllt sind:

1. *Unter allen drei aufeinander folgenden Spalten ist mindestens eine Spalte eine Nullspalte. Mit anderen Worten, für jedes i und jedes positive j gilt $d_{i,j+k} = d_{1-i,j+k} = 0$ für mindestens einem $k \in \{0, \pm 1\}$.*

2. *In keiner Zeile existieren zwei benachbarte Ziffern ungleich Null mit entgegengesetztem Vorzeichen, das heißt der Fall $d_{i,j} \cdot d_{i,j+1} = -1$ tritt niemals ein.*

3. *Falls* $d_{i,j+1} \cdot d_{i,j} \neq 0$ *gilt, dann folgt* $d_{1-i,j+1} = \pm 1$ *und* $d_{1-i,j} = 0$. *Mit anderen Worten, falls in einer Zeile zwei aufeinander folgende Bits ungleich Null sind, ist in der anderen Zeile an der höherwertigen Stelle ein Ziffer ungleich Null und an der niedrigeren Stelle eine Null.*

Algorithmus 17 beschreibt das Verfahren aus [Sol01], mit dem man für zwei positive Ganzzahlen die Joint Sparse Form erhält. Der Algorithmus nutzt dabei die *mods*-Funktion, welche den Rest mit dem kleinsten Absolutwert liefert. In Zeile 11 ist zu

Algorithmus 17 Joint Sparse Form Recoding Methode

Benötigt: Positive Skalare r_0 und r_1.
Liefert: Rekodierte Joint Sparse Form von r_0 und r_1.

1. $j \leftarrow 0$
2. $d_0 \leftarrow 0$
3. $d_1 \leftarrow 0$
4. **while** $r_0 + d_0 > 0$ **or** $r_1 + d_1 > 0$ **do**
5. $\quad l_0 \leftarrow d_0 + r_0$
6. $\quad l_1 \leftarrow d_1 + r_1$
7. $\quad$ **for** $i = 0$ to 1 **do**
8. $\quad\quad$ **if** $l_i \equiv 0 \pmod 2$ **then** {Nächste Ziffer ist eine Null}
9. $\quad\quad\quad u \leftarrow 0$
10. $\quad\quad$ **else**
11. $\quad\quad\quad u \leftarrow l_i \bmod 4$
12. $\quad\quad\quad$ **if** $l_i \equiv \pm 3 \pmod 8$ **and** $l_{i-1} \equiv 2 \pmod 4$ **then** {Anpassung der NAF}
13. $\quad\quad\quad\quad u \leftarrow -u$
14. $\quad\quad\quad$ **end if**
15. $\quad\quad$ **end if**
16. $\quad\quad u_{i,j} \leftarrow u$
17. $\quad$ **end for**
18. $\quad$ **for** $i = 0$ to 1 **do**
19. $\quad\quad$ **if** $2 \cdot d_i = 1 + u_{i,j}$ **then** {Anpassung des Übertrags}
20. $\quad\quad\quad d_i = 1 - d_i$
21. $\quad\quad$ **end if**
22. $\quad\quad r_i \leftarrow \lfloor r_i/2 \rfloor$
23. $\quad$ **end for**
24. $\quad j \leftarrow j + 1$
25. **end while**
26. **return** $\begin{pmatrix} u_{0,j-1} & \dots & u_{0,0} \\ u_{1,j-1} & \dots & u_{1,0} \end{pmatrix}$.

erkennen, dass die JSF-Rekodierungsmethode die NAF-Darstellung nutzt, welche geschickt in Zeile 13 angepasst wird, um möglichst viele Nullspalten zu generieren. Um die Idee dahinter zu verdeutlichen werden folgende Bitkombinationen separat in die ent-

sprechende NAF überführt:

$$\begin{matrix}101\\010\end{matrix}\mapsto\begin{matrix}101\\010\end{matrix}\qquad\begin{matrix}101\\110\end{matrix}\mapsto\begin{matrix}0101\\10\bar{1}0\end{matrix}\qquad\begin{matrix}011\\010\end{matrix}\mapsto\begin{matrix}10\bar{1}\\010\end{matrix}\qquad\begin{matrix}011\\110\end{matrix}\mapsto\begin{matrix}010\bar{1}\\10\bar{1}0\end{matrix}$$

Bei dieser separaten NAF-Rekodierung erhält man keine einzige Nullspalte. In der oberen Zeile wird nun jeweils die Ziffer ganz rechts nicht mit der errechneten NAF-Ziffer, sondern mit der inversen NAF-Ziffer rekodiert. Sie wird folglich nicht durch die Ziffer u sondern durch $-u$ ersetzt. Ansonsten bleibt die NAF-Rekodierung bestehen, das Ergebnis ist dann Folgendes:

$$\begin{matrix}101\\010\end{matrix}\mapsto\begin{matrix}11\bar{1}\\010\end{matrix}\mapsto\begin{matrix}10\bar{1}\bar{1}\\0010\end{matrix}\qquad\begin{matrix}101\\110\end{matrix}\mapsto\begin{matrix}11\bar{1}\\110\end{matrix}\mapsto\begin{matrix}10\bar{1}\bar{1}\\10\bar{1}0\end{matrix}$$

$$\begin{matrix}011\\010\end{matrix}\mapsto\begin{matrix}011\\010\end{matrix}\mapsto\begin{matrix}011\\010\end{matrix}\qquad\begin{matrix}011\\110\end{matrix}\mapsto\begin{matrix}011\\110\end{matrix}\mapsto\begin{matrix}0011\\10\bar{1}0\end{matrix}$$

Diese kleine Anpassung bewirkt, dass in allen Kombinationen eine Nullspalte vorzufinden ist. In Zeile 12 wird genau nach diesen Bitkombinationen gesucht; die eine Bitfolge ist kongruent zu $\pm 3 \pmod 8$ und die gegenüberliegende Bitfolge ist kongruent zu $2 \pmod 4$.

Entsprechend der Definition verwendet die Joint Sparse Form die Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ und stellt somit eine vorzeichenbehaftete Binärdarstellung dar. Solinas hat bewiesen, dass jedes Ganzzahlpaar r_0,r_1 eine eindeutige JSF hat, die höchstens ein Bit länger ist als die längste Binärdarstellung von r_0 und r_1. Des weiteren wurde gezeigt, dass die durchschnittliche joint Hamming-Dichte der JSF

$$\mathcal{AJHD}_2\,(\mathrm{JSF}) = \frac{1}{2}$$

ist, welche für alle gemeinsamen (joint) Bitfolgen in der vorzeichenbehafteten Binärdarstellung das Minimum darstellt [Sol01].

Beispiel 23. *Die zwei Ganzzahlen $r_0 = 51$ und $r_1 = 166$ ergeben in der joint Binärdarstellung die Matrix*

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix},$$

mit nur zwei Nullspalten und somit gilt $\mathcal{JHW}(r_0,r_1) = 6$. Für beide Zahlen in der NAF ergibt sich die Matrix

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & \bar{1} & 0 & 1 & 0 & \bar{1} \\ 1 & 0 & 1 & 0 & 1 & 0 & \bar{1} & 0 \end{pmatrix},$$

mit einem joint Hamming-Gewicht von 8. *Eine Rekodierung der Zahlen* r_0,r_1 *mittels Algorithmus* 17 *hat die Matrix*

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & \bar{1} & 0 & 0 & 1 & 1 \\ 1 & 0 & \bar{1} & 0 & \bar{1} & \bar{1} & 0 & \bar{1} & 0 \end{pmatrix}$$

als Ergebnis mit drei Nullspalten und $\mathcal{JHW}(r_0,r_1) = 6$

5.4.2. Verallgemeinerte Joint Sparse Form

Die Joint Sparse Form von Solinas in [Sol01] ist nur für zwei Ganzzahlen definiert, Proos hat diese in [Pro03] für beliebig viele Ganzzahlen verallgemeinert.

Definition (Joint Sparse Form)**.** *Seien* $r_0,r_1,\ldots,r_{k-1}$ *Ganzzahlen und* n *die maximale Bitlänge der* r_i*'s, dann ist die Matrix*

$$\begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{k-1} \end{pmatrix} = \begin{pmatrix} d_{0,n} & d_{0,n-1} & \ldots & d_{0,0} \\ d_{1,n} & d_{1,n-1} & \ldots & d_{1,0} \\ \vdots & \vdots & & \vdots \\ d_{k-1,n} & d_{k-1,n-1} & \ldots & d_{k-1,0} \end{pmatrix}$$

genau dann in einer Joint Sparse Form, wenn die folgenden Bedingungen alle erfüllt sind:

1. *Für jede Spalte* j*, die keine Nullspalte ist, existiert eine Zeile* $(d_{i,n},d_{i,n-1},\ldots,d_{i,0})$ *mit*

   - $d_{i,j} \neq 0$ *und*
   - *entweder* $j = 0$ *oder es gibt ein* $b < j$ *so dass* $d_{i,j-1} = d_{i,j-2} = \ldots = d_{i,b} = 0$ *und entweder* $b = 0$ *oder die Spalte* b *ist eine Nullspalte.*

2. *In keiner Zeile gibt es zwei aufeinander folgende Bits der Form* $1\bar{1}$ *oder* $\bar{1}1$*.*

3. *Falls eine Zeile* $(d_{i,n},d_{i,n-1},\ldots,d_{i,0})$ *und die Zahlen* j,b *mit den folgenden Eigenschaften* $j > b$*,* $d_{i,j+1} \neq d_{i,j}$ *und* $d_{i,j} = d_{i,j-1} = \ldots = d_{i,b} \neq 0$ *existiert, dann ist die Spalte* $(j+1)$ *eine Nullspalte.*

Entsprechend der Definition ist die Ziffernmenge der JSF $\mathcal{D} = \{0, \pm 1\}$ und ist somit eine Binärdarstellung mit Vorzeichen. Proos hat gezeigt, dass für beliebige k Ganzzahlen eine eindeutige Joint Sparse Form existiert, welche höchstens ein Bit länger ist als die Binärdarstellung der größten Ganzzahl. In [Pro03] wird ein effizienter Algorithmus

vorgestellt, der die JSF erzeugt. Dennoch ist dieser recht umfangreich und kompliziert, daher wird hier nur die Idee des Verfahrens vorgestellt.

Der Kerngedanke, der hinter der JSF-Generierung steht, beruht auf folgender Beobachtung. Für die Binärdarstellung $(0,d_{n-1},d_{n-2},\ldots,d_0)$ einer ungeraden Ganzzahl $d < 2^n$ existiert eine vorzeichenbehaftete Binärdarstellung $(s_n,s_{n-1},\ldots,s_0)$ von d mit $s_{n-1} = 0$, also einer Nullziffer an der Stelle s_{n-1}. Falls die Ziffer d_{n-1} schon eine Null ist, ist die gesuchte Darstellung gerade die Ausgangsdarstellung und es gilt die Zuordnung

$$(s_n,s_{n-1},\ldots,s_0) \mapsto (0,d_{n-1},d_{n-2},\ldots,d_0)\,.$$

Für den Fall, dass $d_{n-1} = 1$ gilt, erhält man durch die Zuordnung

$$(s_n,s_{n-1},\ldots,s_0) \mapsto (1,d_{n-1}-1,d_{n-2}-1,\ldots,d_1-1,\bar{1})$$

die gesuchte Darstellung.

Diese Vorgehensweise lässt sich auch für die vorzeichenbehaftete Darstellung $(s_n,s_{n-1},\ldots,s_0)$ einer beliebigen Ganzzahl $d < 2^n$ verallgemeinern, wobei für einige Ganzzahlen j die Bedingung $s_m \in \{0,1\}$ für $m = j,\ldots,n$ gelten muss. Mit anderen Worten müssen infolgedessen einige Ziffern vom höchstwertigen Bit aus durchgehend vorzeichenfrei sein. Mit einer ähnlichen Umformung, wie oben beschrieben, ist es dann möglich, für alle $b < n$, falls entweder $s_b = 0$ gilt oder ein w mit $j \leq w < b$ und $s_w = 1$ existiert, eine vorzeichenbehaftete Binärdarstellung $(s'_n,s'_{n-1},\ldots,s'_0)$ von d mit folgenden Eigenschaften zu erstellen:

1. $s'_m = s_m$ für $0 \leq m < j$,
2. $s'_b = 0$ und
3. $s'_m \in \{0,1\}$ für $m \geq b+1$.

Solch eine Umformung einer Binärdarstellung mit Vorzeichen nennt Proos eine elementare Umformung. Eine elementare Umformung kann mit den folgenden Zuordnungen erstellt werden. Ist die Ziffer s_b schon eine Null muß nichts geändert werden:

$$(s'_n,s'_{n-1},\ldots,s'_0) \mapsto (s_n,s_{n-1},\ldots,s_0)\,.$$

Falls für die Ziffer $s_b = 1$ gilt, wird ein w mit $j \leq w < b$ und $s_w = 1$ gewählt, sodass die Zuordnungen

$$\begin{aligned}(s'_n,s'_{n-1},\ldots,s'_{b+1}) &\mapsto (s_n,s_{n-1},\ldots,s_{b+1}) + 1\\ (s'_b,s'_{b-1},\ldots,s'_{w+1},s'_w) &\mapsto (0,s_{b-1}-1,\ldots,s_{w+1}-1,\bar{1})\\ (s'_{w-1},s'_{w-2},\ldots,s'_0) &\mapsto (s_{w-1},s_{w-2},\ldots,s_0)\end{aligned}$$

die gewünschte Umformung ergeben. Da die Ganzzahl d kleiner als 2^n ist, hat die vorzeichenbehaftete Binärdarstellung höchstens n-Bits, aus diesem Grund kann eine elementare Umformung niemals länger als $(n+1)$-Bits werden und somit nie über die Ziffer s_n hinaus laufen. Zusammenfassend liefert eine elementare Umformung eine Nullziffer an der Stelle b, wobei alle Bits an den Stellen 0 bis $w-1$ unverändert bleiben und alle Ziffern an den höherwertigen Stellen als b nur positive Vorzeichen haben.

Beispiel 24. *Die Bitfolge* (01011011101) *ist die Binärdarstellung von* $d = 733$ *mit einer vorangestellten Nullziffer. Die folgende Tabelle zeigt ein Beispiel von elementaren Umformungen, die nacheinander von links nach rechts ausgeführt werden. Die Ziffer* d_b *ist hierbei fettgedruckt und* j *gibt die kleinste Zahl an, die die Bedingung* $s_m \in \{0,1\}$ *für* $m = j,\ldots,n$ *erfüllt.*

d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	j	b	w
0	1	0	1	1	0	1	1	**1**	0	1	0	2	0
0	1	0	1	**1**	1	0	0	0	$\bar{1}$	$\bar{1}$	2	6	5
0	**1**	1	0	0	$\bar{1}$	0	0	0	$\bar{1}$	$\bar{1}$	6	9	8
1	0	$\bar{1}$	0	0	$\bar{1}$	0	0	0	$\bar{1}$	$\bar{1}$			

In [Pro03] stellt der Autor eine verallgemeinerte elementare Umformung vor, mit der auch für Ganzzahlen in der vorzeichenbehafteten Binärdarstellung eine Joint Sparse Form erzeugt werden kann. Da die Skalare jedoch typischerweise in der Binärdarstellung vorliegen, reicht die vorgestellte vereinfachte elementare Umformung aus. Der Algorithmus für die Entwicklung einer JSF aus der Ausgangsmatrix

$$T = \begin{pmatrix} 0 & d_{0,n-1} & \ldots & d_{0,0} \\ 0 & d_{1,n-1} & \ldots & d_{1,0} \\ \vdots & \vdots & & \vdots \\ 0 & d_{k-1,n-1} & \ldots & d_{k-1,0} \end{pmatrix},$$

mit den Binärdarstellungen der Skalare und einer vorangestellten Nullspalte, läuft nun folgendermaßen ab. Die Ausgangsmatrix T besteht aus den Zeilen $t_i = (t_{i,n}, t_{i,n-1}, \ldots, t_{i,0})$ und den Spalten $C_n, C_{n-1}, \ldots, C_0$. Der Algorithmus sucht nun von rechts nach links beginnend bei $j = 0$ den kleinsten Block an Spalten $C_r, C_{r-1}, \ldots, C_j$ in T, bei dem eine elementare Umformung in jeder Zeile t_i mit $b = r$ möglich ist. Ist solch ein Block an Spalten gefunden, wird in jeder Zeile eine elementare Umformung mit $b = r$ durchgeführt, die Spalte C_r wird dabei zur Nullspalte. Danach wird die Variable j auf den Wert $j = r+1$ gesetzt und der Algorithmus sucht den nächsten kleinsten Block an Spalten beginnend bei j. Der Algorithmus sucht somit jeweils das kleinste r, welches die folgenden Bedingungen erfüllt:

1. $r \geq j$ und
2. für jedes i gilt entweder $t_{i,r} = 0$ oder es gibt ein z mit $j \leq z < r$ und $t_{i,z} = 1$.

Nach diesen elementaren Umformungen für alle möglichen Spaltenblöcke erfüllt die Matrix die erste Bedingung der JSF-Definition und besitzt zudem die kleinste mögliche Anzahl an Spalten, die keine Nullspalten sind. Damit die rekodierte Matrix auch eine eindeutige Darstellung ergibt, müssen auch die beiden anderen Bedingungen der Definition gelten. Dazu werden am Ende des Algorithmus folgende Änderungen von rechts nach links an der Matrix vorgenommen:

1. In jeder Zeile werden die Bitfolgen $1\bar{1}$ und $\bar{1}1$ durch 01 beziehungsweise $0\bar{1}$ ersetzt.

2. Falls eine Zeile $(d_n,d_{n-1},\ldots,d_0)$ und Ganzzahlen j,b mit $j > b$ existieren, sodass $d_{j+1} = 0$, die Spalte $j+1$ keine Nullspalte ist und $d_j = d_{j-1} = \ldots = d_b = 1$ gilt, wird $(d_{j+1},d_j,\ldots,d_b)$ durch die Bitfolge $(10\ldots0\bar{1})$ ersetzt. Falls $d_j = d_{j-1} = \ldots = d_b = \bar{1}$ gilt, wird $(d_{j+1},d_j,\ldots,d_b)$ entsprechend durch die Bitfolge $(\bar{1}0\ldots01)$ ersetzt.

Durch diese abschließenden Änderungen wird die erste Bedingung der Definition nicht verletzt und es werden keine erzeugten Nullspalten zerstört. Nach diesen Änderungen ist die Matrix in einer eindeutigen Joint Sparse Form gegeben[Pro03].

Die AJHD von k Ganzzahlen in der Joint Sparse Form lässt sich durch die Formel

$$\mathcal{AJHD}_k\,(\mathrm{JSF}) = 1 - \frac{1}{c_k}$$

berechnen, wobei c_k durch die rekursive Formel

$$c_k = \frac{1}{2^k}\left(3 + \sum_{i=1}^{k-1}\binom{k}{i}(c_i + 1)\right)$$

mit $c_1 = 1{,}5$ gegeben ist. In [Pro03] hat Proos bewiesen, dass das joint Hamming-Gewicht für k Skalare in der Joint Sparse Form minimal ist. Die Minimalität der AJHD dieser Darstellung wird hierbei mit eingeschlossen. Unter allen $\mathcal{D}$-Darstellungen mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ existiert folglich keine Darstellung mit geringerer AJHD als die JSF. Tabelle 4 zeigt einige Beispielwerte für die AJHD der JSF von k Ganzzahlen.

k	$\mathcal{AJHD}_k\,(\mathrm{JSF})$	
1	1/3	$\approx 0{,}3333$
2	1/2	$= 0{,}5$
3	23/39	$\approx 0{,}5897$
4	115/179	$\approx 0{,}6424$
5	4279/6327	$\approx 0{,}6763$
6	152821/218357	$\approx 0{,}6998$

Tabelle 4: Beispiele für die $\mathcal{AJHD}_k\,(\mathrm{JSF})$ für k Ganzzahlen

In dieser ist zu erkennen, dass die AJHD für eine Ganzzahl dieselbe ist wie für die NAF-Darstellung. Für $k = 2$ wird das Ergebnis der JSF nach Solinas im vorhergehenden Kapitel bestätigt.

Beispiel 25. *Abschließend soll ein Beispiel mit den Ganzzahlen* $r_0 = 659$,$r_1 = 813$,$r_2 = 626$ *und* $r_3 = 685$ *den Algorithmus zur Entwicklung einer JSF verdeutlichen. Der kleinste mögliche Block von Spalten, in der eine elementare Umformung in jeder Zeile möglich ist, ist hierbei fettgedruckt.*

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & \mathbf{0} & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & \mathbf{0} & 0 & \bar{1} & \bar{1} \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & \mathbf{0} & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & \mathbf{0} & 0 & \bar{1} & \bar{1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \bar{1} & \bar{1} \\ 0 & 1 & 0 & 0 & 1 & \mathbf{1} & \mathbf{1} & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \bar{1} & \bar{1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} \\ 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & \bar{1} & 0 & 0 & 1 & 0 \\ 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & \bar{1} & \bar{1} & 0 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} \\ 0 & 1 & 0 & 1 & 0 & 0 & \bar{1} & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} \end{pmatrix}$$

Nach den elementaren Umformungen sind keine zusätzlichen Änderungen in der Matrix nötig, alle Bedingungen der JSF werden bereits erfüllt. Das joint Hamming-Gewicht beträgt sieben und es wurden genau vier Nullspalten erzeugt.

5.4.3. Entwicklung der Joint Sparse Form von links nach rechts

In [DOT05b] haben die Autoren eine Möglichkeit vorgestellt, mit der die Joint Sparse Form von k Ganzzahlen auch von links nach rechts entwickelt werden kann, die Darstellung wird deshalb als eine left-to-right Joint Sparse Form (ltrJSF) bezeichnet. Die Idee

ist, die Ganzzahlen $r_0,r_1,\ldots,r_{k-1}$ in der MOF-Darstellung in die Matrix

$$\begin{pmatrix} \mathrm{MOF}\,(r_0) \\ \mathrm{MOF}\,(r_1) \\ \vdots \\ \mathrm{MOF}\,(r_{k-1}) \end{pmatrix} = \begin{pmatrix} \mu_{0,n} & \mu_{0,n-1} & \cdots & \mu_{0,0} \\ \mu_{1,n} & \mu_{1,n-1} & \cdots & \mu_{1,0} \\ \vdots & \vdots & & \vdots \\ \mu_{k-1,n} & \mu_{k-1,n-1} & \cdots & \mu_{k-1,0} \end{pmatrix}$$

zu schreiben und die MOF-Eigenschaften auszunutzen, um von links nach rechts Nullspalten zu erzeugen. Dieser Gedanke beruht auf der Beobachtung, dass für jede MOF-Darstellung $(\mu_n,\mu_{n-1},\ldots,\mu_0)$ einer Ganzzahl d mit $0 < d < 2^n$ eine vorzeichenbehaftete Binärdarstellung $(d_n,d_{n-1},\ldots,d_0)$ von d existiert, sodass $d_b = 0$ ist für alle $b \in \{0,\ldots,n\} \setminus \{l\}$. Hierbei ist l die Stelle der Ziffer mit dem niedrigsten Stellenwert in der MOF von d, die ungleich Null ist, dass heißt $\mu_l \neq 0$ und $\mu_i = 0$ für alle $i = 0,1,\ldots,l-1$. Falls die Ziffer μ_b bereits Null ist, muss nichts geändert werden und die gesuchte Darstellung wird durch die Zuweisung

$$(d_n,d_{n-1},\ldots,d_0) \mapsto (\mu_n,\mu_{n-1},\ldots,\mu_0)$$

erhalten. Gilt $\mu_b \neq 0$, wird ein $w \in \{b-1,b-2,\ldots l\}$ gewählt, sodass

$$(\mu_b,\mu_{b-1},\ldots,\mu_{w+1},\mu_w)$$

entweder der Bitfolge $(1,0,\ldots,0,\bar{1})$ oder $(\bar{1},0,\ldots,0,1)$ entspricht. Aufgrund der ersten Bedingung der MOF-Definition existiert solch ein w immer. Mit Hilfe der folgenden Zuordnungen wird die gesuchte Darstellung erzeugt:

$$\begin{aligned} (d_n,d_{n-1},\ldots,d_{b+1}) &\mapsto (\mu_n,\mu_{n-1},\ldots,\mu_{b+1}) \\ (d_b,d_{b-1},\ldots,d_w) &\mapsto (0,\mu_b,\ldots,\mu_b) \\ (d_{w-1},d_{w-2},\ldots,d_0) &\mapsto (\mu_{w-1},\mu_{w-2},\ldots,\mu_0) \end{aligned}$$

Die Umformung ist auf den Bereich $(\mu_b,\mu_{b-1},\ldots,\mu_w)$ der MOF beschränkt, dieser Bereich wird anschaulich entweder durch die Zuordnung

$$\begin{aligned} &(1,0,\ldots,0,\bar{1}) \mapsto (0,1,\ldots,1,1) \text{ oder} \\ &(\bar{1},0,\ldots,0,1) \mapsto (0,\bar{1},\ldots,\bar{1},\bar{1}) \end{aligned}$$

ersetzt, bei der kein Übertrag zustande kommen kann. Zudem bleiben die höherwertigen und niedrigeren Stellen unverändert. Aus diesem Grund ist es möglich, die Umformungen von links nach rechts ohne einen möglichen Übertrag durchzuführen [DOT05b].

Beispiel 26. *Die Ganzzahl $d = 1269$ hat die MOF $(1\bar{1}01000\bar{1}1\bar{1}1\bar{1})$, welche nun von links nach rechts an den möglichen Stellen umgeformt wird. Die Ziffer μ_b ist hierbei jeweils fettgedruckt.*

d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	b	w
1	$\bar{1}$	0	1	0	0	0	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	11	10
0	1	0	**1**	0	0	0	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	8	4
0	1	0	0	1	1	1	1	**1**	$\bar{1}$	1	$\bar{1}$	3	2
0	1	0	0	1	1	1	1	0	1	**1**	$\bar{1}$	1	0
0	1	0	0	1	1	1	1	0	1	0	1		

Der Algorithmus, der die ltrJSF für k Ganzzahlen entwickelt, läuft nun folgendermaßen ab. Die Ganzzahlen werden in ihrer MOF-Darstellung in eine Matrix der Form

$$\begin{pmatrix} \mu_{1,n} & \mu_{1,n-1} & \cdots & \mu_{1,0} \\ \mu_{2,n} & \mu_{2,n-1} & \cdots & \mu_{2,0} \\ \vdots & \vdots & & \vdots \\ \mu_{k,n} & \mu_{k,n-1} & \cdots & \mu_{k,0} \end{pmatrix}$$

geschrieben, die einzelnen Spalten der Matrix werden mit C_n,C_{n-1},...,C_0 bezeichnet. Der Algorithmus sucht nun von links nach rechts beginnend bei $a = n$ den kleinsten Block an Spalten C_a,C_{a-1},...,C_r in der Matrix, bei dem eine Umformung in jeder Zeile an der gleichen Stelle $b \in \{a,a-1,\ldots,r\}$ möglich ist. Ist solch ein Block an Spalten gefunden, wird in jeder Zeile ein Umformung an der Stelle b durchgeführt, die Spalte C_b wird dabei zur Nullspalte. Danach wird die Variable a auf den Wert $a = r-1$ gesetzt und der Algorithmus sucht beginnend bei C_a den nächsten kleinsten Block an Spalten. Der Algorithmus sucht demzufolge jeweils das größte r, welches die folgenden Bedingungen erfüllt:

1. $r \leq a$ und

2. es existiert ein $b \in \{a,a-1,\ldots,r\}$, so dass in jeder Zeile $j = 1,\ldots,k$ entweder

   - $\mu_{j,b} = 0$ gilt oder

   - es existiert ein $w_j \in \{b-1,b-2,\ldots,l_j\}$, so dass $\mu_{j,b} = -\mu_{j,w_j}$ und $\mu_{j,i} = 0$ für $i = b-1,b-2,\ldots,w_j+1$ gilt. Hierbei ist $l_j \in \{a,a-1,\ldots,r\}$ die Stelle der nicht-Null-Ziffer in Zeile j mit dem niedrigsten Stellenwert.

Algorithmus 18 beschreibt die Entwicklung der ltrJSF von k Ganzzahlen in ihren MOF-Darstellungen im Detail. Dabei bezeichnet $Z(\mu)$ die Menge

$$Z\left(\mu_{n-1},\mu_{n-2},\ldots,\mu_0\right) = \{0,1,\ldots,n-1\} \setminus \{l\}$$

für eine n-Bit MOF-Darstellung μ ungleich Null, wobei l die Stelle der nicht-Null-Ziffer mit dem niedrigsten Stellenwert in μ darstellt. Die Schnittmenge von $Z(\mu_j)$ für k n-Bit

Algorithmus 18 left-to-right Joint Sparse Form Recoding Methode [Dah05]

Benötigt: k n-Bit Skalare r_j in deren MOF $(\mu_{j,n},\mu_{j,n-1},\ldots,\mu_{j,0})$, $j = 1,\ldots,k$.
Liefert: ltrJSF-Darstellung $(d_{j,n},d_{j,n-1},\ldots,d_{j,0})$ der k Skalare, $j = 1,\ldots,k$.

1. $a \leftarrow n$
2. **while** $a \geq 0$ **do**
3. $\quad r \leftarrow a$
4. $\quad Z \leftarrow \emptyset$
5. $\quad$ **while** $Z = \emptyset$ and $a \geq 0$ **do**
6. $\quad\quad Z \leftarrow \bigcap_{j=1}^{k} Z\,(\mu_{j,a},\mu_{j,a-1},\ldots,\mu_{j,r})$
7. $\quad\quad$ **if** $Z \neq \emptyset$ **then**
8. $\quad\quad\quad b \leftarrow \max\{z|z \in Z\}$
9. $\quad\quad\quad$ **for** $j = 1$ to k **do**
10. $\quad\quad\quad\quad$ **if** $\mu_{j,b} = 0$ **then**
11. $\quad\quad\quad\quad\quad (d_{j,a},d_{j,a-1},\ldots,d_{j,r}) \leftarrow (\mu_{j,a},\mu_{j,a-1},\ldots,\mu_{j,r})$
12. $\quad\quad\quad\quad$ **else**
13. $\quad\quad\quad\quad\quad (d_{j,a},d_{j,a-1},\ldots,d_{j,b+1}) \leftarrow (\mu_{j,a},\mu_{j,a-1},\ldots,\mu_{j,b+1})$
14. $\quad\quad\quad\quad\quad (d_{j,b},d_{j,b-1},\ldots,d_{j,w_j}) \leftarrow (0,\mu_{j,b},\ldots,\mu_{j,b})$
15. $\quad\quad\quad\quad\quad (d_{j,w_j-1},d_{j,w_j-2},\ldots,d_{j,r}) \leftarrow (\mu_{j,w_j-1},\mu_{j,w_j-2},\ldots,\mu_{j,r})$
16. $\quad\quad\quad\quad$ **end if**
17. $\quad\quad\quad$ **end for**
18. $\quad\quad$ **else**
19. $\quad\quad\quad r \leftarrow r - 1$
20. $\quad\quad$ **end if**
21. $\quad$ **end while**
22. $\quad a \leftarrow r - 1$
23. **end while**
24. **return** $\begin{pmatrix} d_{1,n} & d_{1,n-1} & \ldots & d_{1,0} \\ d_{2,n} & d_{2,n-1} & \ldots & d_{2,0} \\ \vdots & \vdots & & \vdots \\ d_{k,n} & d_{k,n-1} & \ldots & d_{k,0} \end{pmatrix}$.

MOF-Darstellungen wird folglich mit

$$\bigcap_{j=1}^{k} Z\left(\mu_{j,n-1},\mu_{j,n-2},\ldots,\mu_{j,0}\right)$$

bezeichnet. Durch diese Schnittmenge wird in Zeile 6 die Spalte b bestimmt, die zu einer Nullspalte umgeformt werden kann oder gegebenenfalls schon eine Nullspalte ist. Sollte die Schnittmenge mehr als ein Element b enthalten, wird in Zeile 8 das größte von ihnen ausgewählt. Ist der Schnitt leer, kann keine Umformung im aktuellen Spaltenblock durchgeführt werden und der Block wird daraufhin um eine weitere Spalte erweitert. Der Spaltenblock muss dabei für k Skalare höchstens $k+1$ Spalten umfassen, um solch ein b zu finden. Bei k Skalaren gibt es nämlich k mögliche Spalten in denen eine nicht-Nullziffer mit dem niedrigsten Stellenwert stehen kann, aus diesem Grund existiert solch ein b immer innerhalb $k+1$ Spalten. Diese Vorgehensweise im Algorithmus gewährleistet die Eindeutigkeit der ltrJSF, da jeweils der kleinste mögliche Block an Spalten umgeformt und die Nullspalte möglichst weit links erzeugt wird.

Die left-to-right Joint Sparse Form stellt eine vorzeichenbehaftete Binärdarstellung mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ dar. Die Autoren haben in [DOT05b] bewiesen, dass sich die AJHD von k Ganzzahlen in der left-to-right Joint Sparse Form durch die Formel

$$\mathcal{AJHD}_k\left(\text{JSF}\right) = 1 - \frac{1}{c_k}$$

bestimmen lässt, wobei c_k durch die rekursive Formel

$$c_k = \frac{1}{2^k}\left(3 + \sum_{i=1}^{k-1}\binom{k}{i}\left(c_i + 1\right)\right)$$

mit $c_1 = 1{,}5$ gegeben ist. Die AJHD der ltrJSF ist also mit der durchschnittlichen joint Hamming-Dichte der verallgemeinerten JSF für k Ganzzahlen identisch. Der große Vorteil liegt aber in der Möglichkeit, die ltrJSF von links nach rechts zu entwickeln. Tabelle 5 zeigt einige Beispielwerte für die AJHD der ltrJSF von k Ganzzahlen. In

k	$\mathcal{AJHD}_k$ (ltrJSF)	
1	1/3	$\approx 0{,}3333$
2	1/2	$= 0{,}5$
3	23/39	$\approx 0{,}5897$
4	115/179	$\approx 0{,}6424$
5	4279/6327	$\approx 0{,}6763$
6	152821/218357	$\approx 0{,}6998$

Tabelle 5: Beispiele für die $\mathcal{AJHD}_k$ (ltrJSF) für k Ganzzahlen

dieser ist zu erkennen, dass die AJHD für eine Ganzzahl dieselbe ist wie für die 2MOF-Darstellung [DOT05b, Dah05].

In [HKPR05] haben die Autoren bewiesen, dass das joint Hamming-Gewicht für k Skalare in der left-to-right Joint Sparse Form minimal ist, was die Minimalität der AJHD für diese mit einschließt. Unter allen $\mathcal{D}$-Darstellungen mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ existiert folglich keine Darstellung mit geringerer AJHD als die ltrJSF.

Beispiel 27. *Algorithmus 18 entwickelt für die Ganzzahlen* $r_1 = 1253$,$r_2 = 1818$,$r_3 = 154$ *und* $r_4 = 981$ *folgende ltrJSF. Der kleinste mögliche Block von Spalten, bei dem eine Umformung möglich ist, ist hierbei jeweils durch zwei vertikale Linien gekennzeichnet. Die Spalte b, die zur Nullspalte wird, ist fettgedruckt.*

$$\left(\begin{array}{ccc|ccccccccc} 1 & \bar{1} & \mathbf{0} & 1 & 0 & 0 & \bar{1} & 0 & 1 & \bar{1} & 1 & \bar{1} \\ 1 & 0 & \mathbf{0} & \bar{1} & 0 & 0 & 1 & 0 & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 0 & \mathbf{0} & 1 & \bar{1} & 0 & 1 & 0 & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 1 & \mathbf{0} & 0 & 0 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} \end{array}\right)$$

$$\left(\begin{array}{ccc|ccc|ccccc} 1 & \bar{1} & 0 & \mathbf{1} & 0 & 0 & \bar{1} & 0 & 1 & \bar{1} & 1 & \bar{1} \\ 1 & 0 & 0 & \bar{\mathbf{1}} & 0 & 0 & 1 & 0 & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} & \bar{1} & 0 & 1 & 0 & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 1 & 0 & \mathbf{0} & 0 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} \end{array}\right)$$

$$\left(\begin{array}{ccccccc|cc|ccc} 1 & \bar{1} & 0 & 0 & 1 & 1 & 1 & \mathbf{0} & 1 & \bar{1} & 1 & \bar{1} \\ 1 & 0 & 0 & 0 & \bar{1} & \bar{1} & \bar{1} & \mathbf{0} & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{0} & \bar{1} & 1 & \bar{1} & 0 \\ 0 & 1 & 0 & 0 & 0 & \bar{1} & 1 & \bar{\mathbf{1}} & 1 & \bar{1} & 1 & \bar{1} \end{array}\right)$$

$$\left(\begin{array}{ccccccccc|c|c} 1 & \bar{1} & 0 & 0 & 1 & 1 & 1 & 0 & 1 & \bar{\mathbf{1}} & 1 & \bar{1} \\ 1 & 0 & 0 & 0 & \bar{1} & \bar{1} & \bar{1} & 0 & \bar{1} & \mathbf{1} & \bar{1} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \bar{1} & \mathbf{1} & \bar{1} & 0 \\ 0 & 1 & 0 & 0 & 0 & \bar{1} & 1 & 0 & \bar{1} & \bar{\mathbf{1}} & 1 & \bar{1} \end{array}\right)$$

$$\left(\begin{array}{cccccccccccc} 1 & \bar{1} & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & \bar{1} & \bar{1} \\ 1 & 0 & 0 & 0 & \bar{1} & \bar{1} & \bar{1} & 0 & \bar{1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \bar{1} & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & \bar{1} & 1 & 0 & \bar{1} & 0 & \bar{1} & \bar{1} \end{array}\right)$$

5.4.4. Die angepasste Joint Sparse Form DOT-JSF$_3$ für zwei Ganzzahlen

In [KZZ04] haben die Autoren die Joint Sparse Form für zwei Ganzzahlen um die Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ erweitert, indem die Entwicklung der JSF mit einer window-Methode und einer Fensterbreite von drei kombiniert wurde. Die Entwicklung erfolgt dabei vom niedrigsten Bit aus, also von rechts nach links. Die resultierende Darstellung wird mit JSF$_3$ bezeichnet und liefert für beliebige drei aufeinander folgende Spalten mindestens eine und für beliebige fünf aufeinander folgende Spalten mindestens zwei Nullspalten. Die durchschnittliche joint Hamming-Dichte für zwei Ganzzahlen in der

JSF_3 beträgt

$$\mathcal{AJHD}(\text{JSF}_3) = \frac{121}{326} \approx 0{,}3712,$$

was deutlich geringer ist als die entsprechende Dichte der JSF oder ltrJSF mit einer $\mathcal{AJHD}$ von 1/2.

Einen ähnlichen Ansatz verfolgen auch die Autoren in [DOT05a]. Die Entwicklung der JSF für zwei Ganzzahlen wird hierbei mit einer sliding-window-Methode mit mehreren möglichen Fensterbreiten zwischen eins und fünf kombiniert. Die Ganzzahlen werden während der Entwicklung in die MOF-Darstellung überführt, um die MOF-Eigenschaften ausnutzen zu können, folglich ist die Rekodierung von links nach rechts möglich. Die Ziffernmenge der resultierenden Darstellung ist $\mathcal{D} = \{0, \pm 1, \pm 3\}$ und da die Autoren Dahmen, Okeya und Takagi keinen Namen für diese Joint Sparse Form angegeben haben, wird diese hier mit DOT-JSF_3 bezeichnet. In [DOT05a] wird ein effizienter Algorithmus für die DOT-JSF_3 vorgestellt, dennoch ist dieser recht umfangreich und kompliziert. Aus diesem Grund werden hier nur die Kerngedanken, die zu diesem Algorithmus geführt haben, näher erläutert.

Die zwei Ganzzahlen werden in die MOF-Darstellung überführt, um die Rekodierung vom höchstwertigen Bit aus durchführen zu können. Des weiteren werden die Eigenschaften der MOF ausgenutzt, um das joint Hamming-Gewicht zu reduzieren. Die erste MOF-Eigenschaft besagt, dass zwei benachbarte nicht-Null Ziffern unterschiedliche Vorzeichen haben, auch wenn sie durch Nullen getrennt sind. Für w zusammenhängende Ziffern ergeben somit die Bitfolgen

$$(\underbrace{1,0,\ldots,0,\bar{1}}_{w}) \quad \text{und} \quad (\underbrace{\bar{1},0,\ldots,0,1}_{w})$$

den höchsten absoluten Wert mit $2^{w-1}-1$, zusätzliche nicht-Null Ziffern in diesen Bitfolgen reduzieren zwangsläufig diesen Wert. Aus diesem Grund lassen sich alle w zusammenhängende Ziffern mit $w-1$ Nullziffern und einer nicht-Nullziffer mit einem Absolutbetrag von höchstens $2^{w-1}-1$ folgendermaßen darstellen:

$$(\underbrace{0,\ldots,0}_{w-1},2^{w-1}-1) \quad \text{oder} \quad (\underbrace{0,\ldots,0}_{w-1},\overline{2^{w-1}-1}).$$

Sollte in den w Ziffern nur eine Ziffer ungleich Null enthalten sein, existieren ohnedies schon $w-1$ Nullziffern. Wählt man $w = 3$ ergibt das folgende Menge an möglichen Ziffern $\mathcal{D} = \{0, \pm 1, \pm 3\}$, erweitert man die oben genannten Beobachtungen für zwei Skalare ergibt das folgendes Lemma.

Lemma 1. *Für zwei MOF-Darstellungen können mit einer sliding-window Methode und der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ höchstens zwei aufeinander folgende Nullspalten erzeugt werden. Nach dieser Nullspalte muss mindestens eine Spalte ungleich Null folgen.*

Mit Hilfe von Lemma 2 können die möglichen Stellen der Nullspalten ermittelt werden.

Lemma 2. *Seien μ_0 und μ_1 zwei n-Bit lange MOF-Darstellungen. Die Stelle f_0 sei die niedrigste Ziffer ungleich Null in μ_0 und f_1 entsprechend die niedrigste nicht-Null Ziffer in μ_1. Die Menge*

$$Z := \{n-1, n-2, \ldots, 1, 0\} \setminus \{f_0, f_1\}$$

enthält dann die Stellen der Spalten, die zu Nullspalten rekodiert werden können.

Falls $f_0 = f_1$ gilt, müssen folglich für die Erzeugung von zwei Nullspalten, mindestens drei Spalten untersucht werden. Sind f_0 und f_1 unterschiedlich müssen für zwei Nullspalten höchstens vier Spalten untersucht werden.

Beispiel 28. *Zur Verdeutlichung zwei kleine Beispiele, in denen zwei Nullspalten erzeugt werden sollen.*

$$\begin{array}{ccc} 1 & 0 & \bar{1} \\ 0 & 1 & \bar{1} \end{array} \mapsto \begin{array}{ccc} 0 & 0 & 3 \\ 0 & 0 & 1 \end{array} \qquad \begin{array}{cccc} 1 & 0 & \bar{1} & 0 \\ 1 & \bar{1} & 1 & \bar{1} \end{array} \mapsto \begin{array}{cccc} 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & \bar{1} \end{array}$$

$$f_0 = f_1 \quad Z = \{2,1\} \setminus \{0\} \qquad f_0 \neq f_1 \quad Z = \{3,2\} \setminus \{1,0\}$$

Für die Entwicklung der DOT-JSF$_3$ wird die sliding-window Methode mit einer variablen Fensterbreite verwendet. Dabei werden die Breite des Fensters und die Anzahl der gesuchten Nullspalten in diesem Fenster jeweils so gewählt, dass das rekodierte Fenster eine möglichst geringe joint Hamming-Dichte (JHD) hat. Es wird also zuerst versucht, ein Fenster mit einer geringen resultierenden JHD zu rekodieren, sollte dies nicht möglich sein, wird schrittweise eine höhere resultierende JHD akzeptiert. Tabelle 6 stellt die Abfolge der gewählten Fensterbreiten und gesuchten Nullspalten und die daraus resultierende JHD für dieses Fenster dar.

Reihenfolge der Rekodierungsversuche	1.	2.	3.	4.
Benötigte Nullspalten	1	2	3	2
Fensterbreite	1	3	5	4
Resultierende JHD	0	0,33	0,4	0,5

Tabelle 6: Abfolge der Fensterbreiten bei der DOT-JSF$_3$-Entwicklung

Der Algorithmus rekodiert nun Spaltenweise von links nach rechts die zwei MOF-Darstellungen folgendermaßen:

1. Falls die Erzeugung einer Nullspalte bei Fensterbreite $w = 1$ möglich ist, schreibe rekodierte Spalte in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht, weiter bei Punkt 2.

2. Falls die Erzeugung von zwei Nullspalten bei Fensterbreite $w = 3$ möglich ist, schreibe rekodierte Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht, weiter bei Punkt 3.

3. Falls die Erzeugung von drei Nullspalten bei Fensterbreite $w = 5$ möglich ist, schreibe rekodierte Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht, weiter bei Punkt 4.

4. Rekodiere die vier Spalten und erzeuge zwei Nullspalten und

   - falls die letzten zwei Spalten nach der Rekodierung unverändert sind, schreibe die ersten beiden rekodierten Spalten in die Ausgabe. Weiter bei Punkt 1 mit der dritten Spalte.

   - falls die letzte Spalte geändert wurde, diese aber keine Ziffern aus $\{\pm 3\}$ enthält, schreibe die ersten drei rekodierten Spalten in die Ausgabe. Weiter bei Punkt 1 mit der vierten Spalte.

   - falls keine der beiden Bedingungen gilt, schreibe alle vier rekodierten Spalten in die Ausgabe. Weiter bei Punkt 1 mit der nächsten Spalte.

In Punkt 4 wird unter bestimmten Bedingungen die letzte rekodierte Spalte im Fenster in Punkt 1 wiederverwendet. In diesem Fall kann aber nicht mehr garantiert werden, dass zwei benachbarte nicht-Null Ziffern verschiedene Vorzeichen haben. Aus diesem Grund muss Lemma 1 angepasst werden, zu

Lemma 3. *Wird eine bereits rekodierte Spalte wiederverwendet, kann mit einer sliding-window Methode und der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ höchstens eine Nullspalte erzeugt werden. Nach dieser Nullspalte muss mindestens eine Spalte ungleich Null folgen.*

Entsprechend Lemma 2 müssen bei einer wiederverwendeten Spalte folglich mindestens zwei aber höchstens drei Spalten untersucht werden, bis eine Nullspalte erzeugt werden kann.

Beispiel 29. *Zur Verdeutlichung zwei kleine Beispiele, in denen die erste Spalte schon rekodiert wurde und somit zwei benachbarte nicht-Null Ziffern gleiche Vorzeichen haben.*

$$\begin{array}{cc} \bar{1} & \bar{1} \\ 0 & \bar{1} \end{array} \mapsto \begin{array}{cc} 0 & \bar{3} \\ 0 & \bar{1} \end{array} \qquad \begin{array}{ccc} \bar{1} & \bar{1} & 0 \\ 1 & 0 & 1 \end{array} \mapsto \begin{array}{ccc} 0 & \bar{3} & 0 \\ 0 & 3 & \bar{1} \end{array}$$

$$f_0 = f_1 \quad Z = \{1\} \setminus \{0\} \qquad f_0 \neq f_1 \quad Z = \{2\} \setminus \{1{,}0\}$$

Wird eine rekodierte Spalte wiederverwendet, wird im Algorithmus die Abfolge der Fensterbreiten und gesuchten Nullspalten entsprechend angepasst. Tabelle 7 stellt die angepasste Abfolge und die daraus resultierende JHD für die nochmalige Verwendung einer

rekodierten Spalte dar. Diese geänderte Abfolge gilt nur für den Durchlauf mit einer bereits rekodierten Spalte, wird danach keine Spalte wiederverwendet gilt wieder die Abfolge in Tabelle 6.

Reihenfolge der Rekodierungsversuche	1.	2.	3.	4.
Benötigte Nullspalten	1	1	2	1
Fensterbreite	1	2	4	3
Resultierende JHD	0	0,5	0,5	0,66

Tabelle 7: Abfolge der Fensterbreiten bei der nochmaligen Verwendung einer rekodierten Spalte während der DOT-JSF$_3$-Entwicklung

Mit Hilfe der Lemmata ist es nun möglich, für eine gegebene Fensterbreite die Anzahl der möglichen Nullspalten, die erzeugt werden können, zu bestimmen. Lemma 2 gibt Auskunft darüber, an welchen Stellen prinzipiell keine Nullspalten möglich sind. Mit der Menge Z erhält man alle möglichen Stellen. Lemma 1 besagt, dass nach zwei erzeugten Nullspalten eine Spalte ungleich Null folgen muss, und nach Lemma 3 muss bei einer Wiederverwendung schon nach einer erzeugten Nullspalte eine Spalte ungleich Null folgen. Die möglichen Stellen werden dabei von links nach rechts gemäß der beiden Lemmata aus der Menge Z bestimmt. Das Ergebnis ist die Menge $\hat{Z}$ mit den Stellen, an denen nach der Rekodierung die Nullspalten zu finden sind.

Beispiel 30. *Die Fensterbreite sei* $w = 5$, $\mu_0 = \bar{1}01\bar{1}1$ *und* $\mu_1 = 1\bar{1}1\bar{1}1$. *Somit gilt* $Z = \{4,3,2,1\} \setminus \{0\}$ *und nach Lemma 1 erhält man die Nullspalten an den Stellen* $\hat{Z} = \{4,3,\not{2},1\} = \{4,3,1\}$, *da nach zwei Nullspalten eine Spalte ungleich Null folgen muss. Es können insgesamt also drei Nullspalten in diesem Fenster erzeugt werden.*

Beispiel 31. *Sei* $w = 4$, $\mu_0 = \bar{1}\bar{1}\bar{1}1$, $\mu_1 = \bar{1}0\bar{1}1$ *und die erste Spalte wird wiederverwendet. Somit gilt* $Z = \{3,2,1\} \setminus \{0\}$ *und nach Lemma 3 erhält man die Nullspalten an den Stellen* $\hat{Z} = \{3,\not{2},1\} = \{3,1\}$, *da nach einer Nullspalte eine Spalte ungleich Null folgen muss. Es können insgesamt also zwei Nullspalten in diesem Fenster erzeugt werden.*

Wurden die möglichen Nullspalten für eine Fensterbreite im Algorithmus bestimmt und in der Menge $\hat{Z}$ alle Stellen vermerkt, die Null werden sollen, kann der obere und untere Teilstring im Fenster separat rekodiert werden. Dazu wird von links nach rechts nach einer nicht-Null Ziffer gesucht, an deren Stelle eine Null erzeugt werden soll. Wurde solch eine Stelle gefunden, wird rechts davon nach der nächsten nicht-Null Ziffer gesucht und eine der folgenden Umformungen durchgeführt.

$$
\begin{aligned}
&① \quad 100\ldots0\bar{1} \mapsto 011\ldots11 \quad / \quad 1\bar{1} \mapsto 01\\
&② \quad \bar{1}00\ldots01 \mapsto 0\bar{1}\bar{1}\ldots\bar{1}\bar{1} \quad / \quad \bar{1}1 \mapsto 0\bar{1}\\
&③ \quad 100\ldots01 \mapsto 03\bar{1}\ldots\bar{1}\bar{1} \quad / \quad 11 \mapsto 03\\
&④ \quad \bar{1}00\ldots0\bar{1} \mapsto 0\bar{3}1\ldots11 \quad / \quad \bar{1}\bar{1} \mapsto 0\bar{3}
\end{aligned}
$$

Eine weitere nicht-Null Ziffer auf der rechten Seite zu finden ist immer möglich, da durch Lemma 2 die niedrigsten nicht-Null Ziffern der Teilstrings in Z nicht enthalten sind [DOT05a].

Beispiel 32. *Für* $\mu_0 = \bar{1}01\bar{1}1$, $\mu_1 = 1\bar{1}1\bar{1}1$ *und* $\hat{Z} = \{4,3,1\}$ *wird durch die Umformungen* ①-④ *folgende Darstellung erzeugt. Die Stellen, an denen Nullspalten erzeugt werden sollen, sind hierbei fettgedruckt.*

$$\underline{\mathbf{\bar{1}0}1\bar{1}}1 \overset{②}{\longmapsto} \mathbf{0}\underline{\mathbf{1}\bar{1}\bar{1}}1 \overset{④}{\longmapsto} \mathbf{00}\bar{3}\underline{\mathbf{1}\bar{1}}1 \overset{②}{\longmapsto} \mathbf{00}\bar{3}\mathbf{0}\bar{1}$$

$$\underline{\mathbf{1}\bar{1}}1\bar{1}1 \overset{①}{\longmapsto} \mathbf{01}\underline{\mathbf{1}\bar{1}}1 \overset{③}{\longmapsto} \mathbf{00}3\underline{\mathbf{1}\bar{1}}1 \overset{②}{\longmapsto} \mathbf{00}\bar{3}\mathbf{0}\bar{1}$$

Beispiel 33. *Für* $\mu_0 = \bar{1}\bar{1}\bar{1}1$, $\mu_1 = \bar{1}0\bar{1}1$ *und* $\hat{Z} = \{3,1\}$ *erhält man durch die Umformungen* ①-④ *folgende Darstellung.*

$$\underline{\mathbf{\bar{1}\bar{1}}\bar{1}}1 \overset{④}{\longmapsto} \mathbf{0}\bar{3}\underline{\mathbf{\bar{1}}1} \overset{②}{\longmapsto} \mathbf{0}\bar{3}\mathbf{0}\bar{1}$$

$$\underline{\mathbf{\bar{1}0}\bar{1}}1 \overset{④}{\longmapsto} \mathbf{0}\bar{3}\underline{\mathbf{1}1} \overset{③}{\longmapsto} \mathbf{0}\bar{3}\mathbf{0}3$$

Der Algorithmus in [DOT05a] für die Entwicklung der DOT-JSF$_3$-Darstellung entspricht einer Rekodierung von links nach rechts und kann daher mit der Evaluationsphase kombiniert werden. Durch die Entwicklung vom höchstwertigen Bit aus müssen zudem nicht die kompletten Bits der Skalare zwischengespeichert werden. Im vorgeschlagenen Algorithmus müssen für jede Ganzzahl nur noch höchstens sechs Bits auf einmal zwischengespeichert werden, da für die Berechnung der MOF bei der maximalen Fensterbreite von fünf genau sechs Bits der Binärdarstellung benötigt werden. Die Autoren haben in [DOT05a] bewiesen, dass die durchschnittliche joint Hamming-Dichte der DOT-JSF$_3$-Darstellung mit

$$\mathcal{AJHD}(\text{DOT-JSF}_3) = \frac{239}{661} \approx 0{,}3615$$

gegeben ist und somit der window-Methode von [KZZ04] überlegen ist.

Beispiel 34. *Ein abschließendes Beispiel mit* $\mu_0 = 10\bar{1}01\bar{1}001\bar{1}010\bar{1}00$ *und* $\mu_1 = 01\bar{1}001\bar{1}$ $010\bar{1}0100\bar{1}$ *soll den kompletten Algorithmus verdeutlichen. Dabei wird das momentane Fenster durch zwei senkrechte Striche gekennzeichnet. Die möglichen Nullspalten, die im*

momentanen Fenster erzeugt werden können, sind fettgedruckt.

$|1|0\bar{1}01\bar{1}001\bar{1}010\bar{1}00$
$|0|1\bar{1}001\bar{1}010\bar{1}0100\bar{1}$

$|\mathbf{1}\mathbf{0}\bar{1}|01\bar{1}001\bar{1}010\bar{1}00$
$|\mathbf{0}\mathbf{1}\bar{1}|001\bar{1}010\bar{1}0100\bar{1}$

$003|\mathbf{0}|1\bar{1}001\bar{1}010\bar{1}00$
$001|\mathbf{0}|01\bar{1}010\bar{1}0100\bar{1}$

$0030|1|\bar{1}001\bar{1}010\bar{1}00$
$0010|0|1\bar{1}010\bar{1}0100\bar{1}$

$0030|\mathbf{1}1\bar{1}0|01\bar{1}010\bar{1}00$
$0010|\mathbf{0}1\bar{1}|010\bar{1}0100\bar{1}$

$0030|\mathbf{1}\bar{\mathbf{1}}0\mathbf{0}1|\bar{1}010\bar{1}00$
$0010|\mathbf{0}\mathbf{1}\bar{1}\mathbf{0}1|0\bar{1}0100\bar{1}$

$003000303\bar{3}|\bar{1}|010\bar{1}00$
$001000101|0|\bar{1}0100\bar{1}$

$003000303\bar{3}|\bar{\mathbf{1}}01|0\bar{1}00$
$001000101|\mathbf{0}\bar{1}0|100\bar{1}$

$003000303\bar{3}|\bar{\mathbf{1}}\mathbf{0}10\bar{1}|00$
$001000101|\mathbf{0}\bar{\mathbf{1}}010|0\bar{1}$

$003000303\bar{3}|\bar{\mathbf{1}}\mathbf{0}10|\bar{1}00$
$001000101|\mathbf{0}\bar{\mathbf{1}}01|00\bar{1}$

$0030003030\bar{3}00\bar{3}|0|\bar{1}00$
$0010001010\bar{0}\bar{1}|\bar{1}|00\bar{1}$

$0030003030\bar{3}00\bar{3}|0\bar{1}0|$
$0010001010\bar{0}\bar{1}|0\bar{1}0|0\bar{1}$

$0030003030\bar{3}00\bar{3}|\mathbf{0}\bar{1}\mathbf{0}0|$
$0010001010\bar{0}\bar{1}|\bar{\mathbf{1}}0\mathbf{0}\bar{1}|$

$0030003030\bar{3}00\bar{3}0\bar{1}00$
$0010001010\bar{0}\bar{1}0\bar{3}03$

6. Effiziente Berechnung einer zweifachen Punktmultiplikation

In Kapitel 4 wurden zwei effiziente Evaluationsalgorithmen für eine mehrfache Punktmultiplikation vorgestellt, die Shamir Methode und die Interleave Methode. Beide Algorithmen lassen sich wesentlich beschleunigen, wenn die Skalare in eine $\mathcal{D}$-Darstellung mit einer geringen AJHD beziehungsweise einer geringen AHD rekodiert werden. In diesem Kapitel sollen die vorgestellten Rekodierungen aus Kapitel 5 für diese Methoden genutzt werden und beide Methoden für die Evaluation der mehrfachen Punktmultiplikation der Form $k \cdot P + l \cdot Q$ miteinander verglichen werden.

6.1. Beschleunigung der Interleave Methode

Wie in Kapitel 4.2 dargestellt wurde, benötigt die Interleave Methode für die Punktmultiplikation der Form $k \cdot P + l \cdot Q$ im Durchschnitt

$$n\ ECDBL + n \cdot (\mathcal{AHD}(k) + \mathcal{AHD}(l))\ \ ECADD$$

Operationen in der Hauptrechnung. In der Vorberechnungsphase müssen zusätzlich

$$(|\mathcal{D}_1| - 2) + (|\mathcal{D}_2| - 2)$$

Punkte vorberechnet werden. Sind die Skalare in einer vorzeichenbehafteten $\mathcal{D}$-Darstellung mit einer symmetrischen Ziffernmenge der Form $\mathcal{D} = \{0, \pm 1, \ldots, \pm x\}$ gegeben, reduziert sich die Anzahl in der Vorberechnungsphase auf

$$\frac{|\mathcal{D}_1| - 3}{2} + \frac{|\mathcal{D}_2| - 3}{2}$$

Punkte. Bei der Interleave Methode lässt sich für jede Nullziffer in einem der Skalare eine ECADD-Operation einsparen, demnach kann diese Methode vorzugsweise durch eine $\mathcal{D}$-Darstellung mit geringer AHD beschleunigt werden. Obwohl die wNAF und wMOF dieselbe durchschnittliche Hamming-Dichte von

$$\mathcal{AHD}(w\text{NAF}) = \mathcal{AHD}(w\text{MOF}) = \frac{1}{w+1}$$

aufweisen und die identische Ziffernmenge

$$\mathcal{D}_w = \left\{0, \pm 1, \pm 3, \ldots, \pm 2^{w-1} - 1\right\}$$

verwenden, ist die wMOF der wNAF-Darstellung vorzuziehen. Die Interleave Methode arbeitet die Skalare von links nach rechts ab und entspricht dabei der Arbeitsrichtung der wMOF-Rekodierungsmethode. Aus diesem Grund ist es möglich, die Rekodierung „on

the fly“ während der Evaluationsphase durchzuführen. Folglich müssen auch nicht die kompletten rekodierten Skalare zwischengespeichert werden, sondern nur noch die Teile, die gerade für die Evaluation erforderlich sind. Da der wMOF-Algorithmus höchstens w rekodierte Bits auf einmal liefert, müssen bei einer mehrfachen Punktmultiplikation mit zwei Skalaren höchstens $2 \cdot w$ Bits zwischengespeichert werden. Dagegen müssten bei der rechts nach links arbeitenden wNAF-Methode zuerst die kompletten $2 \cdot n$ rekodierten Bits der Skalare zwischengespeichert werden, bevor die Interleave Methode eingesetzt werden könnte. Bei einer mehrfachen Punktmultiplikation mit k Skalaren wird der benötigte Speicherverbrauch von $k \cdot n$ Bits durch die wNAF noch deutlicher bemerkbar. Sind beide Skalare in der wMOF- oder wNAF-Darstellung gegeben, reduziert sich der Aufwand im Durchschnitt für die Interleave Methode auf

$$n\ ECDBL + n \cdot \left(\frac{1}{w_1 + 1} + \frac{1}{w_2 + 1} \right)\ ECADD$$

Operationen in der Hauptrechnung. In der Vorberechnungsphase müssen dazu nur

$$\left(2^{w_1 - 2} - 1\right) + \left(2^{w_2 - 2} - 1\right)$$

Punkte vorberechnet werden, da beide Darstellungen eine symmetrische vorzeichenbehaftete Ziffernmenge aufweisen. Für $w = 2$ müssen überhaupt keine Punkte vorberechnet werden. Sind $w_1 \geq 3$ und $w_2 \geq 3$ werden die vorberechneten Punkte

$$3P,5P,7P,\ldots,(2^{w_1 - 1} - 1)P$$
$$3Q,5Q,7Q,\ldots,(2^{w_2 - 1} - 1)Q$$

in der Hauptrechnung benötigt.

6.2. Beschleunigung der Shamir Methode

Entsprechend Kapitel 4.3 erfordert die Auswertung der mehrfachen Punktmultiplikation $k \cdot P + l \cdot Q$ mittels der Shamir Methode im Durchschnitt

$$n\ ECDBL + n \cdot \mathcal{AJHD}\,(k,l)\ \ ECADD$$

Operationen in der Hauptrechnung. In der Vorberechnungsphase müssen dazu

$$|\mathcal{D}|^2 - 3$$

Punkte bestimmt werden. Sind die Skalare in einer $\mathcal{D}$-Darstellung mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \ldots, \pm x\}$ gegeben, müssen nur noch

$$\frac{|\mathcal{D}|^2 - 1}{2} - 2$$

Punkte in der Vorberechnungsphase bestimmt werden. Bei der Shamir Methode kann für jede Nullspalte in den untereinander geschriebenen Skalaren eine ECADD-Operation eingespart werden, dementsprechend kann die Evaluation durch den Einsatz einer $\mathcal{D}$-Darstellung mit geringer AJHD deutlich beschleunigt werden.

Die JSF und die ltrJSF liefern für zwei Skalare die gleiche durchschnittliche joint Hamming-Dichte von

$$\mathcal{AJHD}(\text{JSF}) = \mathcal{AJHD}(\text{ltrJSF}) = \frac{1}{2},$$

dabei verwenden beide Darstellungen die vorzeichenbehaftete Ziffernmenge

$$\mathcal{D} = \{0, \pm 1\}.$$

Dennoch ist die ltrJSF der JSF vorzuziehen, da erstere während der Evaluationsphase „on the fly“ eingesetzt werden kann. Der Rekodierungsalgorithmus für die ltrJSF liefert für k Ganzzahlen maximal $k \cdot (k+1)$ rekodierte Bits auf einmal, somit müssen für zwei Skalare höchsten sechs rekodierte Bits für die Shamir Methode zwischengespeichert werden. Im Falle der von rechts nach links arbeitenden JSF müssen beide Skalare zuerst komplett rekodiert werden und folglich $2 \cdot n$ Bits für die Evaluation zwischengespeichert werden. Bei der mehrfachen Punktmultiplikation mit mehr als zwei Skalaren macht sich der ineffiezientere Speicherverbrauch durch die JSF noch deutlicher bemerkbar. Treten beide Skalare entweder in der JSF- oder ltrJSF-Darstellung auf, reduziert sich der Aufwand für die Shamir Methode im Durchschnitt auf

$$n\ ECDBL + n \cdot \frac{1}{2}\ ECADD$$

Operationen in der Hauptrechnung. In der Vorberechnungsphase müssen zusätzlich nur die zwei Punkte

$$P + Q \quad \text{und} \quad P - Q$$

bestimmt werden.

Die in Kapitel 5.4.4 vorgestellte DOT-JSF$_3$-Darstellung für zwei Ganzzahlen liefert eine durchschnittliche joint Hamming-Dichte von

$$\mathcal{AJHD}(\text{DOT-JSF}_3) = \frac{239}{661} \approx 0{,}3615$$

und verwendet dabei die symmetrische vorzeichenbehaftete Ziffernmenge

$$\mathcal{D} = \{0, \pm 1, \pm 3\}.$$

Sind die Skalare k und l bei der mehrfachen Punktmultiplikation $k \cdot P + l \cdot Q$ in der DOT-JSF$_3$ dargestellt, reduziert sich der Aufwand für die Shamir Methode auf

$$n\ ECDBL + n \cdot \frac{239}{661}\ ECADD$$

Operationen in der Hauptrechnung. Zusätzlich müssen in der Vorberechnungsphase die zehn Punkte

$$\begin{array}{lllll} 3P & P+Q & P+3Q & 3P+Q & 3P+3Q \\ 3Q & P-Q & P-3Q & 3P-Q & 3P-3Q \end{array}$$

bestimmt werden. Da der Algorithmus zur Entwicklung der DOT-JSF$_3$ die Skalare von links nach rechts abarbeitet, kann dieser als eine „on the fly“ Methode mit der Shamir Methode kombiniert werden. Durch die größte Fensterbreite von fünf in diesem Algorithmus müssen maximal zehn Bits für die Evaluationsmethode zwischengespeichert werden.

Für die zweifache Punktmultiplikation mit nur zwei Skalaren und zwei Punkten ist somit die DOT-JSF$_3$-Darstellung der ltrJSF-Darstellung auf Grund des geringeren Aufwands in der Hauptrechnung vorzuziehen. Für die Evaluation der mehrfachen Punktmultiplikation der Form $\sum_{j=1}^{m} k_j \cdot P_j$ ist die ltrJSF- der allgemeinen JSF-Darstellung überlegen, da sie durch die dynamische Rekodierung einen weit geringeren Speicherverbrauch an den Tag legt.

6.3. Interleave Methode und die Shamir Methode im Vergleich

Vergleicht man den durchschnittlichen Aufwand der Interleave und der Shamir Methode miteinander, ist zu erkennen, dass der Unterschied durch die AHD beziehungsweise der AJHD der Skalare bestimmt wird. Beide Hamming-Dichten sind abhängig von den $\mathcal{D}$-Darstellungsklassen der Skalare und durch die Darstellungsklasse wird wiederum die Anzahl der Punkte bestimmt, die in der Vorberechnungsphase ermittelt werden müssen. Für einen direkten Vergleich zwischen beiden Methoden müssen die Darstellungen so gewählt werden, dass in beiden Vorberechnungsphasen gleich viele Punkte berechnet werden. Im folgenden wird nun der Aufwand zur Berechnung der Punktmultiplikation $k \cdot P + l \cdot Q$ anhand der durchschnittlichen Hammnig-Dichte der Skalare gemessen und für beide Methoden miteinander verglichen.

Werden die Skalare k und l in der 3MOF-Darstellung dargestellt, ist deren Ziffernmenge mit $\mathcal{D} = \{0, \pm 1 \pm 3\}$ gegeben und folglich müssen für eine Evaluation mittels der Interleave Methode die zwei Punkte $\{3P,3Q\}$ vorberechnet werden. Jeder einzelne rekodierte Skalar weist eine AHD von $1/4$ auf und folglich werden durchschnittlich $(\mathcal{AHD}(k) + \mathcal{AHD}(l)) = 1/2$ ECADD-Operationen für jedes Bit in der Hauptrechnung benötigt. Werden für die Skalare die ltrJSF gewählt ist deren Ziffernmenge $\mathcal{D} = \{0 \pm 1\}$ und für die Shamir Methode müssen die zwei Punkte $\{P+Q,P-Q\}$ bestimmt werden. Die AJHD für zwei Skalare in der ltrJSF beträgt $1/2$, demnach ist die durchschnittliche Anzahl der ECADD-Operationen für beide Methoden gleich. Beide Methoden weisen also bei zwei vorzuberechnenden Punkten den selben Aufwand auf.

Der nächstliegende Schritt ist, die Ziffernmenge für die Shamir Methode auf $\mathcal{D} = \{0, \pm 1, \pm 3\}$ zu erweitern, was mit der DOT-JSF$_3$ auch möglich ist. Dadurch wird die Vorberechnung der zehn Punkte $\{3P,3Q,P+Q,P+3Q,3P+Q,3P+3Q,P-Q,P-3Q,3P-Q, 3P-3Q\}$ erforderlich und in der Hauptrechnung werden durchschnittlich nur noch $239/661 \approx 0{,}3615$ ECADD-Operationen benötigt. Damit auch zehn Punkte bei der Interleave Methode vorzuberechnen sind, kann für die Skalare die wMOF-Darstellung mit den zwei unterschiedlichen Fensterbreiten $w = 4$ für den ersten Skalar und $w = 5$ für den zweiten gewählt werden. Daraus ergeben sich die zwei Ziffernmengen $\mathcal{D}_1 = \{0, \pm 1, \pm 3, \pm 5, \pm 7\}$ und $\mathcal{D}_2 = \{0, \pm 1, \pm 3, \ldots, \pm 15\}$, folglich müssen in der Vorberechnungsphase die zehn Punkte $\{3P,5P,7P,3Q,5Q,7Q,9Q,11Q,13Q,15Q\}$ bestimmt werden. Die durchschnittliche Hamming-Dichte für den ersten Skalar ist mit 1/5 und für den zweiten mit 1/6 gegeben, daraus ergibt sich der durchschnittliche Aufwand von $11/30 \approx 0{,}3666$ ECADD-Operationen in der Hauptrechnung. Die zweite Möglichkeit für die Interleave Methode zehn vorzuberechnenden Punkte zu erhalten, wäre die fractional-window-recoding Methode aus Kapitel 5.1.3 einzusetzen. Dazu wählt man für beide Skalare die Ziffernmenge $\mathcal{D} = \{0, \pm 1 \pm 3, \ldots, \pm 11\}$ und erhält somit eine AHD von 2/11 für jedes Skalar. Daraus ergibt sich die Anzahl von $4/11 \approx 0{,}3636$ ECADD-Operationen im Durchschnitt in der Hauptberechnungsphase.

Tabelle 8 fasst den Vergleich zwischen der Interleave- und der Shamir Methode, für die Berechnung von $k \cdot P + l \cdot Q$ zusammen. Alle genannten Rekodierungsmethoden in der Tabelle arbeiten von links nach rechts und können somit mit der Evaluationsphase kombiniert werden. In der Tabelle wird ersichtlich, dass die Shamir Methode durch den Einsatz der DOT-JSF$_3$ und bei zehn vorzuberechnenden Punkten der Interleave Methode überlegen ist [DOT05a].

Verfahren	Vorberechnung	∅ Anzahl von ECADD
Interleave + 3MOF,3MOF	2 Punkte	$\frac{1}{2} \cdot n = 0{,}5000n$
Shamir + ltrJSF	2 Punkte	$\frac{1}{2} \cdot n = 0{,}5000n$
Interleave + 4MOF,5MOF	10 Punkte	$\left(\frac{1}{5} + \frac{1}{6}\right) \cdot n \approx 0{,}3666n$
Interleave + fract. window (m=11)	10 Punkte	$\left(\frac{2}{11} + \frac{2}{11}\right) \cdot n \approx 0{,}3636n$
Shamir + DOT-JSF$_3$	10 Punkte	$\frac{239}{661} \cdot n \approx 0{,}3615n$

Tabelle 8: Vergleich zwischen der Shamir- und der Interleave Methode anhand der durchschnittlichen ECADD-Operationen

7. Fractional Recoding für die Shamir Methode

Wie schon in Kapitel 5.1.3 dargestellt wurde, liefert die fractional-window Rekodierungsmethode einen großen Vorteil. Aufgrund der freien Wahl der Ziffernmenge $\mathcal{D}_m = \{0, \pm 1, \pm 3, \ldots, \pm m\}$ für ein ungerades $m \geq 1$, lassen sich die vorzuberechnenden Punkte für die Evaluationsmethode auf genau $(m-1)/2$ Punkte festlegen. Somit lässt sich der Speicherverbrauch exakt dem zur Verfügung stehenden Speicherplatz anpassen. In Kapitel 6.3 wurde ersichtlich, dass die Shamir Methode in Kombination mit der DOT-JSF_3 und zehn Punkten in der Vorberechnungsphase der Interleave Methode überlegen ist. Nun wäre es interessant, die DOT-JSF_3 so anzupassen, dass die Shamir auch für weniger als zehn vorzuberechnende Punkte der Interleave Methode im direkten Vergleich überlegen ist. Dazu muss eine fractional-window Rekodierung für die Shamir Methode entwickelt werden, und dieses Kapitel soll nun die Grundlagen zur Erstellung solch einer Methode schaffen. Es wird gezeigt, dass solch ein fractional-window Verfahren mit weniger als zehn Punkten möglich ist und die Shamir Methode wiederum der Interleave Methode prinzipiell überlegen sein kann.

7.1. Fractional Recoding für die Shamir Methode mit zehn Punkten

Die Idee der fractional-window Rekodierung für die Shamir Methode (FW-Sh) wird zur Verdeutlichung zuerst mit zehn vorzuberechnenden Punkten erläutert und später für weniger Punkte angepasst. Die zur Verfügung stehende Ziffernmenge ist mit $\mathcal{D} = \{0, \pm 1, \pm 3\}$ gegeben und folglich sind für die Evaluation mit der Shamir Methode zehn Punkte vorzuberechnen. Die zwei Ganzzahlen werden in ihrer MOF-Darstellung untereinander geschriebenen und von links nach rechts abgearbeitet. Der Kerngedanke ist nun mit einer Fensterbreite von $w = 1$ zu starten und durch Umformung wird versucht, eine Nullspalte zu erzeugen. Sollte dies nicht möglich sein, wird die Fensterbreite auf $w = 3$ erhöht, mit dem Ziel zwei Nullspalten zu bilden. Ist dies wiederum nicht möglich, wäre die nächste Stufe der Versuch drei Nullspalten bei $w = 5$ zu bilden und so weiter, mit anderen Worten im x-ten Schritt wird in einem Fenster der Breite $w = (2 \cdot x - 1)$ versucht x Nullspalten zu erzeugen. Wenn dies möglich ist, wird der rekodierte Teil ausgegeben und rechts davon wieder mit einer Fensterbreite von eins gestartet. Tabelle 9 stellt die

Reihenfolge der Rekodierungsversuche	1.	2.	3.	4.	...	x
Benötigte Nullspalten	1	2	3	4	...	x
Fensterbreite	1	3	5	7	...	$2x-1$
Resultierende JHD	0	0,333	0,400	0,428	...	$\frac{x-1}{2x-1}$

Tabelle 9: Abfolge der Fensterbreiten bei der fractional-window Methode für die Shamir Methode

Abfolge der Fensterbreiten, die benötigten Nullspalten für eine erfolgreiche Rekodierung und die daraus resultierende joint Hamming-Dichte für das rekodierte Fenster dar. Das Ziel dieses Algorithmus ist zum einen, eine möglichst geringe AJHD zu erzeugen. Zum anderen soll die durchschnittliche Anzahl an benötigten ECADD-Operationen bei der Evaluation geringer sein als die des entsprechenden fractional-window Verfahrens für die Interleave Methode, welche mit $4/11 \cdot n \approx 0{,}3636n$ gegeben ist. Folglich stellt sich die Frage, wie groß die Schrittweite x gesetzt werden muss, damit die durchschnittliche joint Hamming-Dichte des FW-Sh Verfahrens geringer als dieser Grenzwert wird. Aus diesem Grund wurden alle möglichen Bitkombinationen mit dem obigen Verfahren bis zu einer Fensterbreite x rekodiert, an der die AJHD geringer als 4/11 wird. Tabelle 10 zeigt die Anzahl der Bitkombinationen, die bei gegebener Fensterbreite erfolgreich rekodiert werden konnten und die, bei denen das Fenster erweitert werden musste, da nicht genügend Nullspalten erzeugt werden konnten. Die fünfte Spalte in der Tabelle gibt an, wie groß die Wahrscheinlichkeit ist, die erforderlichen Nullspalten zu erzeugen und wieder bei einer Fensterbreite von eins starten zu können. Die durchschnittliche joint Hamming-Dichte,

x	Fensterbreite	Anzahl rekodiert	Anzahl nicht rekodiert	Rekodierungs-wahrscheinlichkeit	resultierende AJHD
1	1	1	3	0,25	0,75
2	3	24	24	0,5	0,45
3	5	256	128	$0,\overline{6}$	0,38461538461538464
4	7	1536	512	0,75	0,36607142857142855
5	9	6144	2048	0,75	0,3618421052631579
6	11	24576	8192	0,75	0,36080786026200873
7	13	98304	32768	0,75	0,3605507088331516

Tabelle 10: Resultierende AJHD für das fractional-window Verfahren für die Shamir Methode

die bis zum Schritt x erreicht wird, kann mittels des folgenden Quotienten bestimmt werden. Die Anzahl der ausgegebenen nicht-Nullspalten bis zum Schritt x wird geteilt durch die Anzahl der ausgegeben Spalten bis x. In der folgenden Formel bezeichnet pos_x die Wahrscheinlichkeit im Schritt x eine erfolgreiche Rekodierung durchzuführen und neg_x die Wahrscheinlichkeit für eine nicht erfolgreiche. Des weiteren bezeichnet fb_x die aktuelle Fensterbreite im Schritt x und $nonz_x$ gibt an wie viele nicht-Nullspalten bei einer erfolgreichen Rekodierung im gleichen Schritt ausgegeben werden.

$$\mathcal{AJHD}(x) = \frac{\text{Anzahl der nicht-Nullspalten bis } x}{\text{Anzahl der Spalten bis } x} = \frac{nN(x)}{Sp(x)}$$

$$\begin{aligned} nN(x) = pos_1 \cdot nonz_1 + neg_1 \cdot (pos_2 \cdot nonz_2) + neg_1 \cdot neg_2 \cdot (pos_3 \cdot nonz_3) + \\ \ldots + neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot (pos_x \cdot nonz_x) + \\ neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot (neg_x \cdot (nonz_x + 1)) \end{aligned}$$

$$SP(x) = pos_1 \cdot fb_1 + neg_1 \cdot (pos_2 \cdot fb_2) + neg_1 \cdot neg_2 \cdot (pos_3 \cdot fb_3) + \\ \ldots + neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot (pos_x \cdot fb_x) + \\ neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot (neg_x \cdot fb_x)$$

In jedem Schritt x existieren immer Bitkombinationen, die nicht mit der notwendigen Anzahl an Nullspalten umgeformt werden können. Für einen erzwungenen Abbruch am Ende von Schritt x, müssen zuerst alle nicht rekodierten Kombinationen ausgegeben werden, damit auf jeden Fall wieder bei Schritt eins begonnen werden kann. Eine einfache Ausgabe dieser Kombinationen würde einige mögliche Nullspalten verschenken. Daher wurde für alle nicht umgeformten Kombinationen geprüft, ob eine erfolgreiche Rekodierung mit einer Nullspalte weniger als gefordert möglich ist, was für alle Kombinationen bis zum siebten Schritt auch der Fall war.

Beispiel 35. *Bei einer Rekodierung zweier Ganzzahlen mit FW-Sh und einer maximalen Fensterbreite von $w = 3$ ergibt sich eine durchschnittliche joint Hamming-Dichte von* 0,45. *Der Algorithmus ergibt sich dabei wie folgt:*

1. *Falls die Erzeugung einer Nullspalte bei einer Fensterbreite von $w = 1$ möglich ist, schreibe rekodierte Spalte in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht, weiter bei Punkt 2.*

2. *Erweitere das Fenster auf $w = 3$. Falls die Erzeugung von zwei Nullspalten möglich ist, schreibe die rekodierten Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht weiter bei Punkt 3.*

3. *Erzeuge für die Fensterbreite $w = 3$ nur eine Nullspalte, schreibe rekodierte Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort.*

Die Anzahl aller möglichen Bitkombinationen zweier joint MOF-Darstellungen ist in Wahrheit größer als die Tabelle 10 vermuten lässt. Schon im zweiten Schritt gibt es eigentlich 192 Fälle und nicht 48, da es für zwei untereinander geschriebene MOF-Darstellungen genau vier mögliche Vorzeichenvarianten gibt. Jede Variante liefert das gleiche Ergebnis, welches sich nur durch die verschiedenen Vorzeichenvarianten unterscheidet. Durch diesen Umstand ist es ausreichend nur eine der vier Varianten zu betrachten.

Beispiel 36. *Für die 3-Bit breite joint MOF-Darstellung $\begin{smallmatrix}10\bar{1}\\10\bar{1}\end{smallmatrix}$ existieren noch die drei weiteren Vorzeichenvarianten: $\begin{smallmatrix}\bar{1}01\\10\bar{1}\end{smallmatrix}$ $\begin{smallmatrix}\bar{1}01\\\bar{1}01\end{smallmatrix}$ $\begin{smallmatrix}10\bar{1}\\\bar{1}01\end{smallmatrix}$. Die rekodierten Darstellungen unterscheiden sich nur durch die Vorzeichen der einzelnen Ziffern, die Anzahl der erzeugten Nullspalten ist aber bei allen Varianten gleich groß.*

$$\begin{smallmatrix}10\bar{1}\\10\bar{1}\end{smallmatrix} \mapsto \begin{smallmatrix}003\\003\end{smallmatrix} \qquad \begin{smallmatrix}\bar{1}01\\10\bar{1}\end{smallmatrix} \mapsto \begin{smallmatrix}00\bar{3}\\003\end{smallmatrix} \qquad \begin{smallmatrix}\bar{1}01\\\bar{1}01\end{smallmatrix} \mapsto \begin{smallmatrix}00\bar{3}\\00\bar{3}\end{smallmatrix} \qquad \begin{smallmatrix}10\bar{1}\\\bar{1}01\end{smallmatrix} \mapsto \begin{smallmatrix}003\\00\bar{3}\end{smallmatrix}$$

Die Lemmata in Kapitel 5.4.4 bieten die Möglichkeit, die erzeugbaren Nullspalten in der DOT-JSF$_3$ schon vor der eigentlichen Rekodierung bestimmen zu können. Dazu wird im Algorithmus auf Seite 66 eine Bitfolge bestimmt und in einer Variablen z zwischengespeichert. Jede eins in der Bitfolge in z entspricht dabei einer möglichen Nullspalte, die durch eine Umformung des aktuellen Fensters entwickelt werden kann. Demzufolge wird ein Fenster nur dann rekodiert, wenn die erforderliche Anzahl an Nullspalten auch wirklich erzeugt werden kann. Im vorgestellten FW-Sh Verfahren wird die Nullspaltenanzahl erst nach der Rekodierung überprüft und sollte diese zu gering sein, wird die Umformung verworfen und eine alternative Rekodierung versucht. Eine eindeutige Identifizierung der möglichen Nullspalten, die durch eine Umformung entwickelt werden können, konnte für das FW-Sh Verfahren nicht gefunden werden. Die Bitfolge z wurde aber dennoch verwendet. Zum einen, um die Stellen der niedrigsten nicht-Nullziffern der MOF-Darstellungen im Fenster zwischenzuspeichern und zum anderen, um alle möglichen Rekodierungen für das Fenster finden zu können. Dazu wird jedes Fenster im FW-Sh Verfahren für alle möglichen Bitfolgen von z rekodiert und erst danach die erzeugte Nullspaltenanzahl überprüft. Die Bitfolgen müssen dabei genau so viele Einsen aufweisen, wie Nullspalten in dem aktuellen Fenster entwickelt werden sollen. Die eigentlichen Umformungen entsprechen dabei denen der DOT-JSF$_3$-Entwicklung von Seite 69. Das folgende Beispiel soll diese Vorgehensweise verdeutlichen.

Beispiel 37. *Sei $w = 7$ die aktuelle Fensterbreite, dann sind vier Nullspalten für eine erfolgreiche Umformung nötig. Die joint MOF-Darstellung $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}\bar{1}1\end{smallmatrix}$ konnte in den vorhergehenden Schritten nicht erfolgreich rekodiert werden. Die letzte Ziffer ist bei beiden MOFs die niedrigste nicht-Nullziffer, somit ist die letzte Ziffer von z eine Null. Da genau vier Nullspalten gesucht werden, ergeben sich für z die 15 möglichen Bitfolgen mit jeweils vier Einsen:*

1111000 1110100 1110010 1101100 1101010 1100110 1011100 1011010
1010110 1001110 0111100 0111010 0110110 0101110 0011110

Daraus ergeben sich folgende Umformungen:

$\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1111000\end{smallmatrix} \mapsto 0030\bar{3}01$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1110100\end{smallmatrix} \mapsto 003\bar{1}0\bar{1}\bar{1}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1110010\end{smallmatrix} \mapsto 003\bar{1}\bar{1}01$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1101100\end{smallmatrix} \mapsto 0030\bar{3}01$

$\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1101010\end{smallmatrix} \mapsto \mathbf{00}3\mathbf{0}\bar{\mathbf{3}}\mathbf{0}1$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1100110\end{smallmatrix} \mapsto 003\bar{1}00\bar{3}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1011100\end{smallmatrix} \mapsto 0100003\bar{1}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1011010\end{smallmatrix} \mapsto 0100101$

$\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1010110\end{smallmatrix} \mapsto 0101003\bar{3}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 1001110\end{smallmatrix} \mapsto 0100003\bar{1}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 0111100\end{smallmatrix} \mapsto 100\bar{3}0\bar{1}\bar{1}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 0111010\end{smallmatrix} \mapsto 100\bar{3}\bar{1}01$

$\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 0110110\end{smallmatrix} \mapsto \mathbf{100}\bar{\mathbf{3}}\mathbf{00}\bar{\mathbf{3}}$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 0101110\end{smallmatrix} \mapsto 10\bar{1}0\bar{3}01$ $\quad$ $\begin{smallmatrix}1\bar{1}01\bar{1}01\\1000\bar{1}1\\\hline 0011110\end{smallmatrix} \mapsto 1\bar{1}0003\bar{1}$

Nur mit den zwei Bitkombinationen $z = 1101010$ und $z = 0110110$ können vier Nullspalten erzeugt werden, alle anderen Kombinationen schlagen fehl.

In Tabelle 10 wird ersichtlich, dass das FW-Sh Verfahren ab einer Fensterbreite von $w = 9$ mit einer durchschnittlichen joint Hamming-Dichte von etwa 0,3618, dem fractional-window Verfahren für die Interleave Methode mit einer Dichte von ungefähr 0,3636, überlegen ist. Ab einer Fensterbreite von $w = 11$ ist das Verfahren mit AJHD $\approx 0{,}3608$ auch gegenüber der DOT-JSF$_3$ mit AJHD $\approx 0{,}3615$ im Vorteil.

Der Algorithmus zur Entwicklung der DOT-JSF$_3$-Darstellung verfolgt mit der Wiederverwendung von bereits rekodierten Fenstern und steigender Fensterbreite im Grunde den gleichen Ansatz wie das FW-Sh Verfahren. Verfolgt man nämlich bei der DOT-JSF$_3$-Entwicklung alle möglichen Bitkombinationen, ergibt sich insgesamt die gleiche Abfolge der steigenden Fensterbreiten und Nullspalten. Können zum Beispiel bei einer Fensterbreite von $w = 5$ keine drei Nullspalten erzeugt werden, wird das Fenster auf die Breite $w = 4$ verkleinert. Können nun in diesem kleineren Fenster zwei Nullspalten gebildet werden und ist zudem die Wiederverwendung der letzten Spalte möglich und kann im darauf folgenden Schritt im Fenster der Breite $w = 2$ eine Nullspalte erzeugt werden, dann wurden insgesamt drei Nullspalten in einem Fenster der Gesamtbreite von $w = 5$ gebildet. Die Gesamtbreite beträgt nur fünf und nicht sechs, da eine Spalte wiederverwendet wurde. Tabelle 11 zeigt die Anzahl der Bitkombinationen, die bei der resultierenden Gesamtbreite der Fenster erfolgreich mit dem DOT-JSF$_3$-Algorithmus rekodiert werden konnten und die, bei denen das Fenster erweitert werden musste. Die fünfte Spalte in der Tabelle gibt an, wie groß die Wahrscheinlichkeit ist, die erforderlichen Nullspalten erzeugen und wieder bei einer Fensterbreite von eins starten zu können. In

x	Fensterbreite	Anzahl rekodiert	Anzahl nicht rekodiert	Rekodierungs-wahrscheinlichkeit	resultierende AJHD
1	1	1	3	0,25	0,75
2	3	24	24	0,5	0,45
3	5	256	128	$0{,}\overline{6}$	0,38461538461538464
4	7	1392	656	0,68	0,36858258928571
5	9	7488	3008	0,71	0,36540136761468
6	11	34368	13760	0,71	0,36391376140542

Tabelle 11: Resultierende AJHD für die DOT-JSF$_3$-Entwicklung

Tabelle 11 erkennt man, dass im Schritt vier insgesamt weniger Bitkombinationen erfolgreich rekodiert werden können als im FW-Sh Verfahren. Die Ursache hierfür liegt darin, dass die Fensterbreite von sieben nicht im ganzen, sondern in einzelnen und teilweise überlappenden Teilfenstern betrachtet wird. Werden bis zur Fensterbreite $w = 4$ alle Bitmöglichkeiten durchgespielt, erhält man in Punkt vier des DOT-JSF$_3$-Algorithmus von Seite 66 die folgenden Zustände:

- In 24 Fällen können zwei Spalten wiederverwendet werden.
- In 148 Fällen kann eine Spalte wiederverwendet werden.

- In 76 Fällen ist keine Wiederverwendung möglich.

In den 76 Fällen, in denen keine Wiederverwendung möglich ist, werden die vier rekodierten Spalten in die Ausgabe geschrieben und mit der folgenden Spalte bei Punkt 1 des Algorithmus fortgesetzt. In 28 von den 76 Fällen ist die darauf folgende Spalte schon eine Nullspalte, in 24 Fällen eine Spalte der Form $\begin{smallmatrix}1\\0\end{smallmatrix}$ und in den restlichen 24 der Form $\begin{smallmatrix}0\\1\end{smallmatrix}$. In den Fällen, in denen die nachfolgende Spalte keine Nullspalte ist, wird das Fenster auf die Breite $w = 3$ erweitert. Dadurch ergeben sich $48 \cdot 16 = 768$ Kombinationen, in denen zwei Nullspalten erzeugt werden sollen. Diese Kombinationen ergeben sich aus

24 mal

100	**10$\bar{1}$**	**1$\bar{1}$0**	**1$\bar{1}$1**
000	**000**	**000**	**000**
100	**10$\bar{1}$**	1$\bar{1}$0	**1$\bar{1}$1**
001	**001**	001	**001**
100	10$\bar{1}$	**1$\bar{1}$0**	1$\bar{1}$1
010	010	**010**	010
100	**10$\bar{1}$**	1$\bar{1}$0	**1$\bar{1}$1**
01$\bar{1}$	**01$\bar{1}$**	01$\bar{1}$	**01$\bar{1}$**

und 24 mal

000	001	010	01$\bar{1}$
100	100	100	100
000	**001**	010	**01$\bar{1}$**
10$\bar{1}$	**10$\bar{1}$**	10$\bar{1}$	**10$\bar{1}$**
000	001	**010**	01$\bar{1}$
1$\bar{1}$0	1$\bar{1}$0	**1$\bar{1}$0**	1$\bar{1}$0
000	**001**	010	**01$\bar{1}$**
1$\bar{1}$1	**1$\bar{1}$1**	1$\bar{1}$1	**1$\bar{1}$1**

.

Hierbei sind die Fenster, die erfolgreich umgeformt werden können, fettgedruckt. Diese erfolgreichen Umformungen sind in der Ausgabe insgesamt sieben Bit breit, enthalten jeweils vier Nullspalten und sind in Tabelle 11 im Schritt vier enthalten. Genau in diesen 768 Kombinationen liegt der Grund für die unterschiedliche resultierende AJHD der DOT-JSF$_3$ in Tabelle 11 und der FW-Sh in Tabelle 9 ab dem vierten Schritt. Beim fractional-window Verfahren für die Shamir Methode können nämlich zusätzlich die Kombinationen

24 mal

100
001
100
010
100
01$\bar{1}$

und 24 mal

001	010	01$\bar{1}$
100	100	100

erfolgreich umgeformt werden. Dies erklärt auch die Differenz von 144 in der Anzahl der rekodierten Kombinationen im vierten Schritt. Durch eine geschickte Anpassung in der DOT-JSF$_3$-Entwicklung könnte somit die AJHD dieser Darstellung noch weiter verringert werden.

Beispiel 38. *Das folgende Beispiel soll diesen Unterschied in der Entwicklung verdeutlichen. Exemplarisch wird die joint MOF-Darstellung* $\begin{smallmatrix}0100\bar{1}00\\1000001\end{smallmatrix}$ *mit dem DOT-JSF$_3$-*

Algorithmus entwickelt.

$$\begin{array}{l} |0|100\bar{1}00 \\ |1|00000\bar{1} \end{array} \xrightarrow{w=3} \begin{array}{l} |010|0\bar{1}00 \\ |100|000\bar{1} \end{array} \xrightarrow{w=5} \begin{array}{l} |0100\bar{1}|00 \\ |10000|0\bar{1} \end{array} \xrightarrow{w=4}$$

$$\begin{array}{l} |01\mathbf{00}|\bar{1}00 \\ |10\mathbf{00}|00\bar{1} \end{array} \xrightarrow{w=1} \begin{array}{l} 0100|\bar{1}|00 \\ 1000|0|0\bar{1} \end{array} \xrightarrow{w=3} \begin{array}{l} 0100|\bar{1}00| \\ 1000|00\bar{1}| \end{array}$$

Insgesamt können nur zwei Nullspalten gebildet werden. In den letzten zwei Schritten kann keine Nullspalte erzeugt werden, somit können in diesem Fall keine vier Nullspalten bei einer Fenstergesamtbreite von sieben gebildet werden.
Im Gegensatz dazu ist die Bildung von vier Nullspalten sowohl in diesem Fall, als auch für die zwei anderen Varianten in den letzten zwei Spalten, mit dem FW-Sh Verfahren möglich:

$$\begin{array}{l} 0100\bar{1}00 \quad \mathbf{00}1\mathbf{0}3\mathbf{0}0 \\ 100000\bar{1} \mapsto \mathbf{00}3\mathbf{0}3\mathbf{0}3 \\ \overline{1101010} \end{array} \qquad \begin{array}{l} 0100\bar{1}00 \quad \mathbf{00}1\mathbf{0}30\mathbf{0} \\ 10000\bar{1}0 \mapsto \mathbf{00}3\mathbf{0}31\mathbf{0} \\ \overline{1101001} \end{array} \qquad \begin{array}{l} 0100\bar{1}00 \quad \mathbf{00}1\mathbf{0}30\mathbf{0} \\ 10000\bar{1}1 \mapsto \mathbf{00}3\mathbf{0}31\mathbf{0} \\ \overline{1101010} \end{array}$$

7.2. Fractional Recoding für die Shamir Methode mit acht Punkten

Bei Berechnung der skalaren Punktmultiplikation $k \cdot P + l \cdot Q$ mit Hilfe der Interleave Methode und des fractional-window-recoding Verfahrens mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3, \ldots, \pm 9\}$ für die beiden Skalare k und l, müssen in der Vorberechnungsphase die acht Punkte $\{3P,5P,7P,9P,3Q,5Q,7Q,9Q\}$ ermittelt werden. Für jedes Skalar ergibt sich eine durchschnittliche Hamming-Dichte von $4/21$, demnach werden in der Evaluationsphase im Durchschnitt $8/21 \approx 0{,}38095$ ECADD-Operationen ausgeführt. Dieser Wert soll nun mit dem FW-Sh Verfahren und der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ unterboten werden. Dazu müssen aus den zehn vorzuberechnenden Punkten $\{3P,3Q,P+Q,P+3Q,3P+Q,3P+3Q,P-Q,P-3Q,3P-Q,3P-3Q\}$ acht ausgewählt werden, die in der Vorberechnungsphase bestimmt werden. Die zwei übrigen Punkte werden nicht vorberechnet und stehen somit in der Evaluationsphase nicht zur Verfügung und können folglich auch nicht in der Rekodierungsphase verwendet werden. Insgesamt ergeben sich 45 Möglichkeiten für die zwei fehlenden Punkte. Für jede Möglichkeit wurden alle Bitkombinationen bis zu einer Fensterbreite von elf mit dem FW-Sh Verfahren durchgespielt. In Tabelle 17 im Anhang sind die Ergebnisse für alle Möglichkeiten aufgelistet. Diejenigen mit der geringsten resultierenden AJHD wurden auch bis zu einer Breite von $w = 13$ berechnet und sind in der Tabelle 18 im Anhang zu finden. Die besten Ergebnisse sind in Tabelle 12 zusammengefasst. Hier ist zu erkennen, dass das FW-Sh Verfahren in vier Fällen ab einer Breite von $w = 11$ der Interleave Methode überlegen ist. Die geringsten Werte werden mit den zwei fehlenden Punkten $\{3P+Q,3P-Q\}$ oder mit $\{P+3Q,P-3Q\}$ erlangt. In diesen Fällen ergibt sich bei $w = 11$ eine AJHD von ungefähr 0,37963 und bei der nächsthöheren Fensterbreite von $w = 13$ eine AJHD um die 0,37839.

FP	w	pos	neg	pos-1	neg-1	%	AJHD
3P+Q	1	2	6	6	0	0,25	0,75
3P-Q	3	40	56	56	0	0,4166667	0,475
	5	460	436	436	0	0,5133929	0,4149305555555556
	7	4048	2928	2928	0	0,5802752	0,3919578622816033
	9	28832	18016	18016	0	0,6154372	0,3829907975460123
	11	181456	106800	106800	0	0,629496	0,37963383514760424
	13	1091872	616928	616928	0	0,63897	0,37838921528508057
3P+Q	1	2	6	6	0	0,25	0,75
P-3Q	3	40	56	56	0	0,4166667	0,475
	5	460	436	436	0	0,5133929	0,4149305555555556
	7	4032	2944	2944	0	0,5779817	0,3920863309352518
	9	28592	18512	18512	0	0,6069973	0,38325717615309124
	11	182384	113808	113808	0	0,6157627	0,37992528965091227
	13	1128080	692848	692848	0	0,6195083	0,37866328065880883
3P-Q	1	2	6	6	0	0,25	0,75
P+3Q	3	40	56	56	0	0,4166667	0,475
	5	460	436	436	0	0,5133929	0,4149305555555556
	7	4032	2944	2944	0	0,5779817	0,3920863309352518
	9	28592	18512	18512	0	0,6069973	0,38325717615309124
	11	182384	113808	113808	0	0,6157627	0,37992528965091227
	13	1128080	692848	692848	0	0,6195083	0,37866328065880883
P+3Q	1	2	6	6	0	0,25	0,75
P-3Q	3	40	56	56	0	0,4166667	0,475
	5	460	436	436	0	0,5133929	0,4149305555555556
	7	4048	2928	2928	0	0,5802752	0,3919578622816033
	9	28832	18016	18016	0	0,6154372	0,3829907975460123
	11	181456	106800	106800	0	0,629496	0,37963383514760424
	13	1091872	616928	616928	0	0,63897	0,37838921528508057

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalte-1
neg ≙ Anzahl der negativen Rekodierungen, neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalte-1
% ≙ Wahrscheinlickeit für positive Rekodierung

Tabelle 12: Resultierende AJHD bei der fractional-window Methode für die Shamir Methode mit acht vorzuberechnenden Punkten

Da zwei der zehn möglichen Punkte in der Rekodierungsphase nicht mehr zur Verfügung stehen, reicht es nicht mehr aus, nur eine der vier Vorzeichenvarianten für die zwei joint MOF-Darstellungen zu überprüfen. Dennoch müssen nicht alle vier, sondern nur zwei Varianten überprüft werden, weil jeweils zwei Vorzeichenvarianten zueinander komplementär sind. Durch diesen Umstand müssen zum Beispiel im zweiten Schritt mit $w = 3$ nicht alle 192 Fälle überprüft werden, sondern nur die Hälfte.

Beispiel 39. *Für die 3-Bit breite joint MOF-Darstellung* $\begin{smallmatrix}10\bar{1}\\10\bar{1}\end{smallmatrix}$ *existieren noch die drei weiteren Vorzeichenvarianten:* $\begin{smallmatrix}\bar{1}01\\10\bar{1}\end{smallmatrix}$ $\begin{smallmatrix}\bar{1}01\\\bar{1}01\end{smallmatrix}$ $\begin{smallmatrix}10\bar{1}\\\bar{1}01\end{smallmatrix}$*. Fehlt in der Rekodierungsphase der Punkt* $3P - 3Q$*, dann steht auch der Punkt* $-3P + 3Q$ *nicht zur Verfügung. Fehlt einer dieser Punkte in einer Vorzeichenvariante, fehlt der negierte Punkt auch in der komplementären Variante.*

$$\begin{smallmatrix}10\bar{1}\\10\bar{1}\end{smallmatrix} \mapsto \begin{smallmatrix}003\\003\end{smallmatrix} \qquad \begin{smallmatrix}\bar{1}01\\10\bar{1}\end{smallmatrix} \mapsto \begin{smallmatrix}00\not{3}\\00\not{3}\end{smallmatrix} \qquad \begin{smallmatrix}\bar{1}01\\\bar{1}01\end{smallmatrix} \mapsto \begin{smallmatrix}00\bar{3}\\00\bar{3}\end{smallmatrix} \qquad \begin{smallmatrix}10\bar{1}\\\bar{1}01\end{smallmatrix} \mapsto \begin{smallmatrix}00\not{3}\\00\not{3}\end{smallmatrix}$$

Die obere Bitfolge entspricht dabei dem Skalar k *und die untere dem Skalar* l*.*

Die Vorgehensweise des FW-Sh Verfahrens mit nur acht Punkten ist analog der mit zehn aus dem vorigen Kapitel. Eine kleine Anpassung ist aber dennoch nötig, da nach einer Rekodierung nicht nur die Anzahl der Nullspalten überprüft werden muss, sondern auch nach fehlenden Punkten in der rekodierten Darstellung gesucht werden muss. Wird ein fehlender Punkt im rekodierten Fenster gefunden, wird die aktuelle Rekodierung verworfen und nach einem weiteren Bitfolge z gesucht, mit dem eine erfolgreiche Umformung möglich sein könnte. Wurden bereits alle möglichen Varianten von z getestet ist eine erfolgreiche Rekodierung des Fensters nicht möglich. Wie im vorigen Kapitel wurde für jedes Fenster, das nicht erfolgreich umgeformt werden konnte, überprüft ob eine erfolgreiche Umformung mit einer Nullspalte weniger als erforderlich möglich ist. Wie in Tabelle 17 im Anhang zu entnehmen ist, war dies nicht für alle Möglichkeiten von zwei fehlenden Punkten durchgehend gegeben. Für diese Fälle kann auch keine resultierende AJHD angegeben werden und sie sind somit für das FW-Sh Verfahren unbrauchbar, da fehlende Punkte nicht in der rekodierten Darstellung auftauchen dürfen.

Beispiel 40. *Seien die zwei fehlenden Punkte* $3P + Q$ *und* $3P - Q$*. Bei einer Rekodierung zweier Ganzzahlen mit dem FW-Sh Verfahren mit nur acht Punkten und einer maximalen Fensterbreite von* $w = 3$ *wird eine durchschnittliche joint Hamming-Dichte von* 0,475 *erlangt. Der Algorithmus ergibt sich dabei wie folgt:*

1. *Falls die Erzeugung einer Nullspalte bei einer Fensterbreite von* $w = 1$ *möglich ist, schreibe rekodierte Spalte in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht, weiter bei Punkt 2.*

2. *Erweitere das Fenster auf* $w = 3$*. Falls durch Umformung zwei Nullspalten im Fenster erzeugt werden können und kein fehlender Punkt im umgeformten Fenster enthalten ist, schreibe die rekodierten Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort. Falls nicht weiter bei Punkt 3.*

3. *Erzeuge für die Fensterbreite $w = 3$ nur eine Nullspalte, wobei kein fehlender Punkt enthalten sein darf. Schreibe rekodierte Spalten in Ausgabe und fahre mit nächster Spalte bei Punkt 1 fort.*

Bei einer detaillierteren Betrachtung des FW-Sh Verfahrens ist aufgefallen, dass bei größeren Fensterbreiten ab $w = 9$ in einigen Fällen mehr als nur die geforderten Nullspalten erzeugt werden können. Der Grund dafür liegt in den fehlenden Punkten, die für die Umformungen nicht zur Verfügung stehen. Durch diesen Umstand kann es vorkommen, dass die Fensterbreite soweit erhöht werden muss, dass in einem darauf folgendem Schritt eine Nullspalte mehr als gefordert gebildet werden kann. Ein kleines Beispiel dient der Veranschaulichung dieser Sachlage.

Beispiel 41. *Die joint MOF-Darstellung* $\begin{smallmatrix}10\bar{1}0001\bar{1}0\\0010\bar{1}10\bar{1}0\end{smallmatrix}$ *wird mit FW-Sh und den zwei fehlenden Punkten $3P + Q$ und $3P - Q$ rekodiert. Bei einer Fensterbreite von drei könnten durch die Umformung*

$$\begin{matrix}10\bar{1}\\001\end{matrix} \mapsto \begin{matrix}003\\001\end{matrix}$$

zwei Nullspalten gebildet werden, der fehlende Punkt $3P + Q$ macht dies jedoch unmöglich. Aufgrund der fehlenden Punkte muss die Breite des Fensters bis auf neun erhöht werden, bis eine erfolgreiche Umformung gefunden wird. Durch

$$\begin{matrix}10\bar{1}0001\bar{1}0\\0010\bar{1}10\bar{1}0\\\overline{100110101}\end{matrix} \mapsto \begin{matrix}011000010\\00100\bar{1}0\bar{1}0\end{matrix}$$

können die fünf erforderlichen Nullspalten gebildet werden. Mit der folgenden Umformung

$$\begin{matrix}10\bar{1}0001\bar{1}0\\0010\bar{1}10\bar{1}0\\\overline{110101101}\end{matrix} \mapsto \begin{matrix}003000010\\000030030\end{matrix}$$

können aber sogar sechs Nullspalten erzeugt werden.

Für alle Kombinationen an fehlenden Punkten, die für das FW-Sh Verfahren verwendet werden können, wurde in jedem Schritt überprüft, ob auch eine Nullspalte mehr als gefordert gebildet werden kann. Im Anhang in Tabelle 19 sind die einzelnen Ergebnisse aufgelistet, die resultierende AJHD für die einzelnen Fälle ist im Anhang in Tabelle 20 zu finden. Die besten Ergebnisse und die mögliche Verringerung der AJHD durch diese zusätzlichen Nullspalten ist in Tabelle 13 zusammengefasst. In zwei Fällen kann die AJHD bei einer Fensterbreite von $w = 11$ sogar auf ungefähr 0,37954 verringert werden. Im Vergleich dazu ist die geringste AJHD ohne diese zusätzlichen Nullspalten bei der gleichen Fensterbreite mit 0,37963 gegeben. Durch diese kleine Anpassung kann die durchschnittliche joint Hamming-Dichte folglich noch weiter verringert werden.

Zur Berechnung der durchschnittlichen joint Hamming-Dichte, die bis zum Schritt x erreicht wird, muss die Formel für die zusätzlichen Nullspalten angepasst werden. Dazu

FP	w	pos	pos+1	neg	pos-1	AJHD	AJHD verringert um
3P+Q	1	2	0	6	6	0,75	0
3P-Q	3	40	0	56	56	0,475	0
	5	460	0	436	436	0,4149305555555556	0
	7	4048	0	2928	2928	0,3919578622816033	0
	9	28676	156	18016	18016	0,382916027607362	0,0000747699386503
	11	180784	672	106800	106800	0,3795405459397521	0,00009328920785214
3P+Q	1	2	0	6	6	0,75	0
P-3Q	3	40	0	56	56	0,475	0
	5	460	0	436	436	0,4149305555555556	0
	7	4032	0	2944	2944	0,3920863309352518	0
	9	28476	116	18512	18512	0,38320159163395484	0,0000555845191364
	11	181848	536	113808	113808	0,3798549014659575	0,00007038818495477
3P-Q	1	2	0	6	6	0,75	0
P+3Q	3	40	0	56	56	0,475	0
	5	460	0	436	436	0,4149305555555556	0
	7	4032	0	2944	2944	0,3920863309352518	0
	9	28476	116	18512	18512	0,38320159163395484	0,0000555845191364
	11	181848	536	113808	113808	0,3798549014659575	0,00007038818495477
P+3Q	1	2	0	6	6	0,75	0
P-3Q	3	40	0	56	56	0,475	0
	5	460	0	436	436	0,4149305555555556	0
	7	4048	0	2928	2928	0,3919578622816033	0
	9	28676	156	18016	18016	0,382916027607362	0,0000747699386503
	11	180784	672	106800	106800	0,3795405459397521	0,00009328920785214

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, pos+1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten+1
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

Tabelle 13: Resultierende AJHD mit zusätzlichen Nullspalten bei der fractional-window Methode für die Shamir Methode mit acht vorzuberechnenden Punkten

werden in der folgenden Formel mit $posP_x$ in jedem Schritt $x > 1$ die möglichen Fälle berücksichtigt, in denen eine Nullspalte mehr als nötig erzeugt werden kann.

$$\mathcal{AJHD}(x) = \frac{\text{Anzahl der nicht-Nullspalten bis } x}{\text{Anzahl der Spalten bis } x} = \frac{nN(x)}{Sp(x)} \tag{12}$$

$$\begin{aligned} nN(x) = pos_1 \cdot nonz_1 + neg_1 \cdot (pos_2 \cdot nonz_2 + posP_2 \cdot (nonz_2 - 1)) + \\ neg_1 \cdot neg_2 \cdot (pos_3 \cdot nonz_3 + posP_3 \cdot (nonz_3 - 1)) + \ldots + \\ neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot \Big(pos_x \cdot nonz_x + posP_x \cdot (nonz_x - 1) + neg_x \cdot (nonz_x + 1)\Big) \end{aligned}$$

$$\begin{aligned} SP(x) = pos_1 \cdot fb_1 + neg_1 \cdot fb_2 \cdot (pos_2 + posP_2) + neg_1 \cdot neg_2 \cdot fb_3 \cdot (pos_3 + posP_3) + \\ \ldots + neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot \Big(fb_x \cdot (pos_x + posP_x + neg_x)\Big) \end{aligned}$$

7.3. Fractional Recoding für die Shamir Methode mit vier Punkten

Für die Berechnung der Punktmultiplikation $k \cdot P + l \cdot Q$ mit Hilfe der Interleave Methode in Kombination mit der fractional-window Rekodierung und der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3, \pm 5\}$ für k und l müssen in der Vorberechnungsphase die vier Punkte $\{3P,5P,3Q,5Q\}$ bestimmt werden. Durch die Rekodierung ergibt sich für jedes Skalar eine durchschnittliche Hamming-Dichte von $2/9$, infolgedessen werden in der Evaluationsphase im Durchschnitt $4/9 = 0{,}\overline{4}$ ECADD-Operationen verwendet. In diesem Kapitel soll nun überprüft werden, ob dieser Wert durch das FW-Sh Verfahren mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ und der Verwendung von nur vier vorzuberechnenden Punkten unterboten werden kann. Für einen direkten Vergleich dürfen aus den zehn Punkten, die aufgrund der Ziffernmenge in der Evaluationsphase ermittelt werden müssten, insgesamt nur vier bestimmt werden. Demnach stehen in der Rekodierungsphase sechs von zehn möglichen Punkten nicht zur Verfügung, insgesamt ergeben sich demzufolge 210 Möglichkeiten. Für jede einzelne Möglichkeit wurden alle Bitkombinationen bis zu einer Fensterbreite von $w = 11$ mit dem FW-Sh Verfahren durchgespielt, im Anhang in Tabelle 21 sind die einzelnen Ergebnisse aufgelistet.

Bei keiner der 210 Möglichkeiten konnte zuerst ein positives Ergebnis ermittelt werden, da bei allen Möglichkeiten in jedem Schritt Fälle vorkommen, die selbst mit einer Nullspalte weniger nicht erfolgreich umgeformt werden können. Somit konnte in keinem Schritt die Umformung abgebrochen werden, um wieder bei einer Fensterbreite von eins starten zu können. Dies ist auf die große Anzahl an fehlenden Punkten zurückzuführen, die bei der Rekodierung nicht zur Verfügung stehen. Für einen erzwungenen Abbruch im Schritt x musste die Vorgehensweise wie folgt erweitert werden:

Schritt x Versuche im Schritt x im Fenster mit der Breite $2x-1$ durch Umformungen genau x Nullspalten zu bilden. Falls dies möglich ist, schreibe umgeformtes Fenster in Ausgabe und fahre mit nächster Spalte bei Schritt 1 fort. Falls nicht weiter bei Punkt $nsp-1$.

nsp − 1 Falls durch eine erfolgreiche Umformung nur $x-1$ Nullspalten im Fenster mit der Breite $2x-1$ erzeugt werden können, schreibe rekodiertes Fenster in Ausgabe und fahre mit nächster Spalte bei Schritt 1 fort. Falls nicht weiter bei Punkt $nsp-2$.

nsp − 2 Bilde im Fenster durch Umformungen $x-2$ Nullspalten, schreibe das Fenster in Ausgabe und fahre mit nächster Spalte bei Schritt 1 fort.

Mit diesem geänderten Verfahren wurden alle Bitkombinationen bis zu einer Fensterbreite von elf und für einige viel versprechende Fälle sogar bis zu $w = 13$ überprüft. Durch die geänderte Vorgehensweise konnte das FW-Sh Verfahren für drei Fälle angewendet und die resultierende AJHD bestimmt werden. In allen anderen Fällen konnte die Dichte nur geschätzt werden, da immer noch fehlende Punkte bei den Rekodierungen mit zwei Nullspalten weniger auftraten. Bei der Schätzung wurde angenommen, dass die nicht zu rekodierenden Spalten ohne eine Nullspalte ausgegeben werden. Die drei positiven Fälle sind in Tabelle 14 aufgelistet, dabei kann in zwei Fällen der Wert der Interleave Methode mit einer resultierenden AJHD von $\approx 0{,}44094$, bei einer maximalen Fensterbreite von $w = 13$, unterboten werden.

Zur Berechnung der resultierenden durchschnittlichen joint Hamming-Dichte, die bis zum Schritt x erreicht wird, muss die Formel um die zwei Variablen $posM1$ und $posM2$ erweitert werden. Dabei bezeichnet $posM1_x$ die Wahrscheinlichkeit einer erfolgreichen Rekodierung im x-ten Schritt mit einer Nullspalte weniger als nötig. Des weiteren ist $posM2_x$ die Wahrscheinlichkeit für eine erfolgreiche Umformung mit zwei Nullspalten weniger.

$$\mathcal{AJHD}(x) = \frac{\text{Anzahl der nicht-Nullspalten bis } x}{\text{Anzahl der Spalten bis } x} = \frac{nN(x)}{Sp(x)}$$

$$nN(x) = pos_1 \cdot nonz_1 + neg_1 \cdot (pos_2 \cdot nonz_2) + neg_1 \cdot neg_2 \cdot (pos_3 \cdot nonz_3) + \ldots +$$
$$neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot \Big(pos_x \cdot nonz_x + posM1_x \cdot (nonz_x + 1) + posM2_x \cdot (nonz_x + 2)\Big)$$

$$SP(x) = pos_1 \cdot fb_1 + neg_1 \cdot (pos_2 \cdot fb_2) + neg_1 \cdot neg_2 \cdot (pos_3 \cdot fb_3) + \ldots +$$
$$neg_1 \cdot neg_2 \cdot \ldots \cdot neg_{x-1} \cdot \Big(fb_x \cdot (pos_x + posM1_x + posM2)\Big)$$

Für die drei positiven Fälle an fehlenden Punkten wurde auch überprüft, ob wiederum Umformungen möglich sind, die eine Nullspalte mehr als gefordert erzeugen. In nur einem Fall war dies ab einer Fensterbreite von $w = 9$ möglich. Wie in Tabelle 15 ersichtlich

FP	w	pos	neg	pos-1	neg-1	pos-2	%	AJHD
3P+0	1	2	6	6	0	0	0,25	0,75
0+3Q	3	24	72	72	0	0	0,25	0,525
3P+Q	5	284	868	856	12	12	0,2465	0,48060344827586216
3P-Q	7	3380	10508	10292	216	216	0,2434	0,46138100436681223
3P+3Q	9	40416	127712	124400	3312	3312	0,2404	0,45109681577314176
3P-3Q	11	485352	1558040	1509752	48288	48288	0,2375	0,44492055546474885
	13	5852608	19076032	18391408	684624	684624	0,2348	0,4409356093689873
3P+0	1	2	6	6	0	0	0,25	0,75
0+3Q	3	24	72	72	0	0	0,25	0,525
P+3Q	5	284	868	856	12	12	0,2465	0,48060344827586216
P-3Q	7	3380	10508	10292	216	216	0,2434	0,46138100436681223
3P+3Q	9	40416	127712	124400	3312	3312	0,2404	0,45109681577314176
3P-3Q	11	485352	1558040	1509752	48288	48288	0,2375	0,44492055546474885
	13	5852608	19076032	18391408	684624	684624	0,2348	0,4409356093689873
3P+Q	1	2	6	6	0	0	0,25	0,75
3P-Q	3	24	72	72	0	0	0,25	0,525
P+3Q	5	216	936	848	88	88	0,1875	0,49999999999999994
P-3Q	7	2872	12104	10912	1192	1192	0,1918	0,4792383820998279
3P+3Q	9	36744	156920	137872	19048	19048	0,1897	0,46885697104264973
3P-3Q	11	459696	2051024	1758280	292744	292744	0,1831	0,46246004471356833
	13	5762024	27054360	22670352	4384008	4384008	0,1756	0,4582485972285563

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg ≙ Anzahl der negativen Rekodierungen, neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1
pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2, % ≙ Wahrscheinlichkeit für positive Rekodierung

Tabelle 14: Resultierende AJHD bei der fractional-window Methode für die Shamir Methode mit vier vorzuberechnenden Punkten

wird, konnte zwar durch diesen Trick die resultierende AJHD reduziert werden, dennoch konnte der Wert der Interleave Methode bis zu Breite $w = 11$ nicht unterboten werden. Im Anhang sind in Tabelle 22 die Ergebnisse der restlichen Fälle aufgeführt. Aufgrund der Berechnungsdauer von mehreren Wochen für $w = 13$ können leider für diese Fensterbreite keine Angaben gemacht werden. Dennoch ist es recht unwahrscheinlich, dass sich die resultierende AJHD bei $w = 13$ durch die zusätzlichen Nullspalten auf einen Wert unterhalb von $0,\overline{4}$ reduzieren lässt. Zur Berechnung der einzelnen resultierenden durchschnittlichen joint Hamming-Dichten muss die obige Formel, analog zu (12), nur um die zusätzlichen Nullspalten erweitert werden.

Fehlende Punkte = $\{3P + Q, 3P - Q, P + 3Q, P - 3Q, 3P + 3Q, 3P - 3Q\}$

w	pos	pos+1	neg	pos-1	pos-2	AJHD	AJHD verringert um
1	2	0	6	6	0	0,75	0
3	24	0	72	72	0	0,525	0
5	216	0	936	848	88	0,49999999999999994	0
7	2872	0	12104	10912	1192	0,4792383820998279	0
9	36712	32	156920	137872	19048	0,4687966142852188	0,00006035675743093
11	459272	424	2051024	1758280	292744	0,4623973078331728	0,00006273688039553

w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2

Tabelle 15: Resultierende AJHD mit zusätzlichen Nullspalten bei der fractional-window Methode für die Shamir Methode mit vier vorzuberechnenden Punkten

8. Schlussfolgerung

In dieser Arbeit wurden Kryptosysteme mit elliptischen Kurven vorgestellt und gezeigt, dass diese besonders für rechenschwache Systeme, wie Smartcards und PDAs, interessant sind. Aber auch für Systeme, deren Ressourcen nicht beschränkt sind, liefern EC-Kryptosysteme wesentliche Vorteile. Im Vergleich zu Kryptosystemen, deren Sicherheit auf dem Faktorisierungsproblem oder dem diskreten Logarithmusproblem in Primkörpern beruht, können die Schlüssellängen bei den Systemen mit elliptischen Kurven um einiges kürzer gewählt werden. Dadurch ergeben sich zum einen geringere Ausführungszeiten bei der Ver- und Entschlüsselung. Zum anderen müssen die EC-Schlüssel für ein höheres Sicherheitsniveau nur um wenige Bits verlängert werden. Für einen vergleichbaren Sicherheitsgewinn bei einem RSA-Schlüssel muss dieser schon um einige Hundert Bits verlängert werden.

Die Grundvorraussetzung für Kryptosysteme mit elliptischen Kurven ist die Multiplikation eines Skalars mit einem elliptischen Kurvenpunkt. Werden mehrere Skalarmultiplikationen zusammenaddiert spricht man von einer mehrfachen Punktmultiplikation. Die Sicherheit der EC-Systeme beruht auf der Schwierigkeit, aus dem Ergebnis einer skalaren Punktmultiplikation den Skalar zu bestimmen.

Für die Ausführungszeiten von EC-Systemen sind besonders die Berechnungszeiten für die einfache und mehrfache Punktmultiplikation entscheidend, wobei in dieser Arbeit der Schwerpunkt hauptsächlich auf die mehrfache Punktmultiplikation der Form $k \cdot P + l \cdot Q$ gelegt wurde. Mit der Interleave und der Shamir Methode wurden zwei Algorithmen vorgestellt, mit denen die mehrfache Punktmultiplikation effizient berechnet werden kann. Es wurde gezeigt, dass die Effizienz der Interleave Methode weitestgehend von der durchschnittlichen Hamming-Dichte (AHD) der einzelnen Skalare abhängt. Im Gegensatz dazu ist bei der Shamir Methode die durchschnittliche joint Hamming-Dichte (AJHD) der Skalare entscheidend.

Aus diesem Grund wurden effiziente Rekodierungsverfahren vorgestellt, mit denen sich die AJHD und die AHD der Skalare verringern lässt. Der wMOF Rekodierungsalgorithmus liefert eine $\mathcal{D}$-Darstellung eines Skalars mit minimaler durchschnittlicher Hamming-Dichte. Mit Hilfe der fractional-window Rekodierungsmethode erhält man eine weitere vorzeichenbehaftete Darstellung mit minimaler AHD. Mit dieser Methode lassen sich zudem die Anzahl der Punkte, die im Evaluationsalgorithmus vorberechnet werden müssen, exakt festlegen und somit dem zur Verfügung stehenden Speicherplatz anpassen. Eine vorzeichenbehaftete Binärdarstellung mit minimaler AJHD für zwei Skalare liefert das left-to-right JSF Rekodierungsverfahren. Durch das DOT-JSF_3 Verfahren erhält man eine joint Darstellung mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$, die eine noch geringere AJHD aufweist, als die lrtJSF. Alle vier Verfahren haben den großen Vorteil, dass die Skalare von links nach rechts abgearbeitet werden, infolgedessen kann die Rekodierung dynamisch während der Evaluation erfolgen.

Wie sich herausgestellt hat, lassen sich die Evaluationsalgorithmen mit Hilfe der Rekodierungsverfahren signifikant beschleunigen. Ein Vergleich zwischen der Interleave und der Shamir Methode, für die Berechnung der Punktmultiplikation $k \cdot P + l \cdot Q$, hat folgendes aufgezeigt. Werden bei beiden Methoden jeweils zehn Punkte vorberechnet, ist die Shamir Methode mit der DOT-JSF_3 Darstellung der Interleave Methode in Kombination mit der fractional-window Rekodierung eindeutig überlegen.

Verfahren	Vorberechnung	∅ Anzahl von ECADD
Interleave + fract. window (m=11)	10 Punkte	$\approx 0{,}36364n$
Shamir + DOT-JSF_3	10 Punkte	$\approx 0{,}36157n$
FW-Shamir bis $w = 11$	10 Punkte	$\approx 0{,}36081n$
FW-Shamir bis $w = 13$	10 Punkte	$\approx 0{,}36055n$
Interleave + fract. window (m=9)	8 Punkte	$\approx 0{,}38095n$
FW-Shamir mit Nsp+1 bis $w = 11$	8 Punkte	$\approx 0{,}37954n$
FW-Shamir bis $w = 11$	8 Punkte	$\approx 0{,}37963n$
FW-Shamir bis $w = 13$	8 Punkte	$\approx 0{,}37839n$
Interleave + fract. window (m=5)	4 Punkte	$\approx 0{,}44444n$
FW-Shamir bis $w = 13$	4 Punkte	$\approx 0{,}44094n$

Tabelle 16: Vergleich zwischen dem Fractional-window Shamir Verfahren und der Interleave Methode anhand der durchschnittlichen ECADD-Operationen

Im Anschluss wurde gezeigt, dass sich die AJHD der DOT-JSF_3 grundsätzlich durch eine geschickte Anpassung noch ein wenig reduzieren lässt. Die Schwierigkeit besteht darin, eine geschickte Anpassung zu finden, die den Algorithmus nicht ineffizient werden lässt. In zukünftigen Untersuchungen könnte nach dieser Anpassung geforscht werden.

Es wurde dargelegt, dass sich die Vorteile der fractional-window Rekodierung auch mit der Shamir Methode kombinieren lassen. Dazu wurde das fractional-window Rekodierungsverfahren mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ für die Shamir Methode (FW-Sh) mit zehn, acht und vier vorzuberechnenden Punkten vorgestellt. Anhand der durchschnittlich benötigten ECADD-Operationen konnte in allen drei Fällen nachgewiesen werden, dass das FW-Sh Verfahren der fractional-window Interleave Methode überlegen ist. In Tabelle 16 werden die besten Ergebnisse des vorgestellten Verfahrens zusammengefasst und mit der Interleave Methode verglichen. Das FW-Sh Verfahren kann trotz dieser theoretischen Überlegenheit die anderen Verfahren nicht ersetzen, da noch kein effizienter Algorithmus gefunden wurde. Diese Ergebnisse können aber als Grundlage

für weitere Untersuchungen dienen, in denen nach einem effizienten FW-Sh Algorithmus geforscht wird.

Mit der fractional-window Rekodierungsmethode erhält man bewiesenermaßen eine $\mathcal{D}$-Darstellung mit minimaler durchschnittlicher Hamming-Dichte, des Weiteren ist auch die Minimalität der wMOF-Darstellung bewiesen. Folglich können die benötigten ECADD-Operationen in der Interleave Methode nicht weiter reduziert werden, da keine $\mathcal{D}$-Darstellung mit noch geringerer AHD existiert. Für joint Darstellungen mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1\}$ wurde bewiesen, dass keine Darstellung mit geringerer AJHD existiert, als die Joint Sparse Form. Im Gegensatz dazu konnte für joint Darstellungen mit der Ziffernmenge $\mathcal{D} = \{0, \pm 1, \pm 3\}$ noch keine minimale Darstellung gefunden werden. Die DOT-JSF$_3$ stellt ein effizientes Verfahren dar, welches eine AJHD von $239/661 \approx 0{,}36157$ ermöglicht. Mittels dem vorgestellten FW-Sh Verfahren konnte dargelegt werden, dass grundsätzlich eine AJHD von $\approx 0{,}36055$ möglich ist. Aufbauend auf diesen Ergebnissen kann die Entwicklung eines leistungsfähigen Algorithmus in zukünftigen Forschungen erfolgen.

Literatur

[Ava04] Avanzi, Roberto Maria: *A Note on the Signed Sliding Window Integer Recoding and a Left-to-Right Analogue*. In: Handschuh, Helena und M. Anwar Hasan (Herausgeber): *Selected Areas in Cryptography*, Band 3357 der Reihe *Lecture Notes in Computer Science*, Seiten 130–143. Springer, 2004.

[BBF02] Bertsch, Andreas, Frank Bourseau und Dirk Fox: *Perspektive kryptografischer Verfahren auf elliptischen Kurven*. Datenschutz und Datensicherheit, 26(2), 2002.

[BSS05] Blake, Ian F., Gadiel Seroussi und Nigel P. Smart (Herausgeber): *Advances in elliptic curve cryptography*, Band 317 der Reihe *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2005.

[Buc03] Buchmann, Johannes: *Einführung in die Kryptographie (Springer-Lehrbuch)*. Springer, September 2003.

[Cer00] Certicom Research: *Standards For Efficient Cryptography SEC 1: Elliptic Curve Cryptography*, 2000. http://www.secg.org/collateral/sec1_final.pdf.

[Cer02] Certicom Corp: *Notre Dame Mathematician Solves ECCp-109 Encryption Key Problem Issued in 1997*, 2002. http://www.certicom.com/index.php/2002-press-releases/38-2002-press-releases/340-notre-dame-mathematician-solves-eccp-109-encryption-key-problem-issued-in-1997.

[CFA+06] Cohen, Henri, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen und Frederik Vercauteren (Herausgeber): *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.

[Dah05] Dahmen, Erik: *Efficient Algorithms for Multi-Scalar Multiplications*. Diplomarbeit, Technische Universität Darmstadt, 2005.

[DH76] Diffie, W. und M. E. Hellam: *New Directions in Cryptography*. IEEE Transactions on Information Theory, IT-22(6), nov 1976.

[DOT05a] Dahmen, Erik, Katsuyuki Okeya und Tsuyoshi Takagi: *An Advanced Method for Joint Scalar Multiplications on Memory Constraint Devices.* In: Molva, Refik, Gene Tsudik und Dirk Westhoff (Herausgeber): *ESAS*, Band 3813 der Reihe *Lecture Notes in Computer Science*, Seiten 189–204. Springer, 2005.

[DOT05b] Dahmen, Erik, Katsuyuki Okeya und Tsuyoshi Takagi: *Efficient Left-to-Right Multi-Exponentiations.* Technischer Bericht, Technische Universität Darmstadt, 2005.

[Eck07] Eckert, Claudia: *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* Oldenbourg, 5., überarbeitete Auflage. Auflage, November 2007.

[Eng99] Enge, Andreas: *Elliptic curves and their applications to cryptography: an introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[Gam85] Gamal, Taher El: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* In: Blakley, G. R. und D. C. Chaum (Herausgeber): *Proc. CRYPTO 84*, Seiten 10–18. Springer, 1985. Lecture Notes in Computer Science No. 196.

[GN06] Gehrmann, C. und M. Naslund: *Ecrypt Yearly report on algorithms and keysizes (2006).* Technischer Bericht, Ecrypt, 2006. http://www.ecrypt.eu.org/documents/D.SPA.21-1.1.pdf.

[Gor98] Gordon, Daniel M.: *A Survey of Fast Exponentiation Methods.* Journal of Algorithms, 27:129–146, 1998.

[Hen03] Henhapl, Birgit: *Zur Effizienz von Elliptische Kurven Kryptographie.* Doktorarbeit, Technische Universität Darmstadt, 2003.

[HKPR05] Heuberger, Clemens, Rajendra S. Katti, Helmut Prodinger und Xiaoyu Ruan: *The alternating greedy expansion and applications to computing digit expansions from left-to-right in cryptography.* Theoretical Computer Science, 341(1-3):55–72, 2005.

[HMV03] Hankerson, Darrel, Alfred J. Menezes und Scott Vanstone: *Guide to Elliptic Curve Cryptography.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[Hus03] Husemöller, Dale: *Elliptic Curves (Graduate Texts in Mathematics).* Springer, December 2003.

[JMV01] Johnson, Don B., Alfred J. Menezes und Scott Vanstone: *The Elliptic Curve Digital Signature Algorithm (ECDSA)*. International Journal of Information Security, 1(1):36–63, 2001.

[Kob87a] Koblitz, Neal: *A course in number theory and cryptography*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[Kob87b] Koblitz, Neal: *Elliptic Curve Cryptosystems*. Mathematics of Computation, 48(177):203–209, 1987.

[Kob98] Koblitz, Neal: *Algebraic aspects of cryptography*. Springer, 1998.

[KZZ04] Kuang, BaiJie, Yuefei Zhu und YaJuan Zhang: *An Improved Algorithm for uP + vQ Using JSF$_3^1$*. In: Jakobsson, Markus, Moti Yung und Jianying Zhou (Herausgeber): *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, Band 3089 der Reihe *Lecture Notes in Computer Science*, Seiten 467–478. Springer, 2004.

[LL93] Lenstra, A. K. und H. W. Lenstra: *The Development of the Number Field Sieve*, Band 1554 der Reihe *Lecture Notes in Mathematics*. Springer, 1993.

[LL94] Lim, Chae Hoon und Pil Joong Lee: *More Flexible Exponentiation with Precomputation*. In: *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, Seiten 95–107, London, UK, 1994. Springer-Verlag.

[Loc05] Lochter, M.: *ECC Brainpool Standard Curves and Curve Generation, v. 1.0 (19.10.05)*. Technischer Bericht, ECC Brainpool, 2005. http://www.ecc-brainpool.org/download/Domain-parameters.pdf.

[LRW03] Lamparter, B., I. Riedel und D. Westhoff: *Anmerkungen zur Nutzung digitaler Signaturen in Ad Hoc Netzwerken*. PIK - Praxis der Informationsverarbeitung und Kommunikation: Fachzeitschrift für den Einsatz von Informationssystemen., 26(4):223–227, 2003. http://www.iponair.de/publications/Lamparter_PIK03.pdf.

[Mat07] Matzies, Alexander: *Implementierung flexibler Skalarmultiplikation auf Elliptischen Kurven*. Diplomarbeit, Technische Universität Darmstadt, 2007.

[Mil86] Miller, Victor S: *Use of elliptic curves in cryptography.* In: *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, Seiten 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[Möl01] Möller, Bodo: *Algorithms for Multi-exponentiation.* In: *Selected Areas in Cryptography - SAC*, Seiten 165–180. Springer, 2001.

[Möl04] Möller, Bodo: *Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation.* In: Park, Choonsik und Seongtaek Chee (Herausgeber): *ICISC*, Band 3506 der Reihe *Lecture Notes in Computer Science*, Seiten 137–153. Springer, 2004.

[MO90] Morain, F. und J. Olivos: *Speeding up the computations on an elliptic curve using addition-subtraction chains.* RAIRO Technical Informatics and and Applications, 24(6):531–543, 1990.

[MP98] Müller, V. und S. Paulus: *Elliptische Kurven und Public Key Kryptographie.* Technische Universität Darmstadt, 1998. http://www.cdc.informatik.tu-darmstadt.de/reports/reports/vm-sp.dud1.ps.gz.

[MS06] Muir, James A. und Douglas R. Stinson: *Minimality and other properties of the width-w nonadjacent form.* Mathematics of Computation, 75(253):369–384, jan 2006.

[MvOV97] Menezes, Alfred J., Paul C. van Oorschot und Scott A. Vanstone: *Handbook of Applied Cryptography.* CRC Press, 1997.

[Odl85] Odlyzko, A M: *Discrete logarithms in finite fields and their cryptographic significance.* In: *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*, Seiten 224–314, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[OSSST04] Okeya, Katsuyuki, Katja Schmidt-Samoa, Christian Spahn und Tsuyoshi Takagi: *Signed Binary Representations Revisited.* In: Franklin, Matthew K. (Herausgeber): *Advances in Cryptology*, Band 3152 der Reihe *Lecture Notes in Computer Science*, Seiten 123–139. Springer, 2004.

[PH78] Pohlig, S. und M. Hellman: *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.).* Information Theory, IEEE Transactions on, 24(1):106–110, 1978.

[Pol78] Pollard, John M.: *Monte Carlo methods for index computation mod p.* Mathematics of Computation, 32:918–924, 1978.

[Pro03] Proos, John: *Joint Sparse Forms and Generating Zero Columns when Combing.* Technischer Bericht, University of Waterloo, 2003.

[Rei60] Reitwiesner, George W.: *Binary Arithmetic.* Advances in Computers, 1:231–308, 1960.

[RHAL92] Rivest, Ronald L., Martin E. Hellman, John C. Anderson und John W. Lyons: *Responses to NIST's proposal.* Commun. ACM, 35(7):41–54, 1992.

[Sch05] Schneier, Bruce: *Angewandte Kryptographie - Der Klassiker. Protokolle, Algorithmen und Sourcecode in C.* Pearson Studium, 12 2005.

[Sha71] Shanks, Daniel: *Class Number, a Theory of Factorization, and Genera.* In: *Analytic Number Theory*, Band 20 der Reihe *Proceedings of Symposia in Pure Mathematics*, Seiten 415–440. American Mathematical Society, 1971.

[Sil85] Silverman, Joseph H.: *The Arithmetic of Elliptic Curves (Graduate Texts in Mathematics).* Springer, December 1985.

[Sol00] Solinas, Jerome A.: *Efficient Arithmetic on Koblitz Curves.* Designs, Codes, and Cryptography, 19(2–3):195–249, März 2000.

[Sol01] Solinas, Jerome A.: *Low-weight binary representations for pairs of integers.* Technischer Bericht, University of Waterloo, 2001.

[Sti02] Stinson, Douglas R.: *Cryptography: Theory and Practice, Second Edition.* Chapman & Hall/CRC, February 2002.

[Was03] Washington, Lawrence C.: *Elliptic Curves: Number Theory and Cryptography.* Discrete Mathematics and Its Applications. Chapman & Hall/CRC, May 2003.

[WMPW98] Win, Erik De, Serge Mister, Bart Preneel und Michael J. Wiener: *On the Performance of Signature Schemes Based on Elliptic Curves.* In: *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*, Seiten 252–266, London, UK, 1998. Springer-Verlag.

A. Tabellen für die FW-Sh Methode mit acht vorzuberechnenden Punkten

Tabelle 17: Ergebnisse des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten

		Spalten beginnen mit $\begin{smallmatrix}1\\0\end{smallmatrix}$				Spalten beginnen mit $\begin{smallmatrix}0\\1\end{smallmatrix}$				Spalten beginnen mit $\begin{smallmatrix}1\\1\end{smallmatrix}$			
FP	w	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1
P+Q	3	17	15	12	3	17	15	12	3	6	26	14	12
3P-3Q	5	139	101	101	0	139	101	101	0	134	282	196	86
	7	984	632	632	0	984	632	632	0	1806	2706	2216	490
	9	6048	4064	4064	0	6048	4064	4064	0	19230	24066	21112	2954
P+Q	3	16	16	8	8	16	16	8	8	8	24	0	24
P-Q	5	140	116	86	30	140	116	86	30	144	240	16	224
	7	1182	674	536	138	1182	674	536	138	1680	2160	608	1552
	9	7282	3502	2900	602	7282	3502	2900	602	17600	16960	7648	9312
	11	38786	17246	14748	2498	38786	17246	14748	2498	153616	117744	66848	50896
P+Q	3	17	15	12	3	13	19	16	3	10	22	10	12
3P+0	5	125	115	113	2	155	149	145	4	158	194	108	86
	7	1030	810	800	10	1348	1036	1024	12	1496	1608	1078	530
	9	7472	5488	5472	16	9752	6824	6804	20	12658	13070	9942	3128
	11	51370	36438	36396	42	65476	43708	43660	48	102514	106606	87612	18994
P+Q	3	13	19	16	3	17	15	12	3	10	22	10	12
0+3Q	5	155	149	145	4	125	115	113	2	158	194	108	86
	7	1348	1036	1024	12	1030	810	800	10	1496	1608	1078	530
	9	9752	6824	6804	20	7472	5488	5472	16	12658	13070	9942	3128
	11	65476	43708	43660	48	51370	36438	36396	42	102514	106606	87612	18994
P+Q	3	17	15	12	3	13	19	16	3	10	22	10	12
3P+Q	5	143	97	97	0	121	183	183	0	166	186	92	94
	7	1020	532	528	4	1361	1567	1543	24	1478	1498	846	652
	9	5710	2802	2802	0	12949	12123	12095	28	11816	12152	8052	4100
	11	30650	14182	14182	0	107507	86461	86303	158	94870	99562	75718	23844
P+Q	3	17	15	12	3	13	19	14	5	10	22	10	12
3P-Q	5	139	101	101	0	131	173	160	13	174	178	88	90
	7	1024	592	592	0	1515	1253	1198	55	1512	1336	822	514
	9	6308	3164	3164	0	12127	7921	7742	179	11430	9946	7346	2600
	11	34964	15660	15660	0	78881	47855	47189	666	87628	71508	59446	12062
P+Q	3	13	19	16	3	17	15	12	3	10	22	10	12
P+3Q	5	121	183	183	0	143	97	97	0	166	186	92	94
	7	1361	1567	1543	24	1020	532	528	4	1478	1498	846	652
	9	12949	12123	12095	28	5710	2802	2802	0	11816	12152	8052	4100
	11	107507	86461	86303	158	30650	14182	14182	0	94870	99562	75718	23844
P+Q	3	13	19	14	5	17	15	12	3	10	22	10	12
P-3Q	5	131	173	160	13	139	101	101	0	174	178	88	90
	7	1515	1253	1198	55	1024	592	592	0	1512	1336	822	514
	9	12127	7921	7742	179	6308	3164	3164	0	11430	9946	7346	2600
	11	78881	47855	47189	666	34964	15660	15660	0	87628	71508	59446	12062
P+Q	3	17	15	12	3	17	15	12	3	6	26	10	16
3P+3Q	5	143	97	97	0	143	97	97	0	96	320	64	256
	7	1008	544	544	0	1008	544	544	0	656	4464	368	4096
	9	5604	3100	3060	40	5604	3100	3060	40	3720	67704	2080	65624
	11	30856	18744	18320	424	30856	18744	18320	424	21008	1062256	13328	1048928
P-Q	3	17	15	12	3	17	15	12	3	6	26	10	16
3P-3Q	5	143	97	97	0	143	97	97	0	96	320	64	256
	7	1008	544	544	0	1008	544	544	0	656	4464	368	4096
	9	5604	3100	3060	40	5604	3100	3060	40	3720	67704	2080	65624
	11	30856	18744	18320	424	30856	18744	18320	424	21008	1062256	13328	1048928
P-Q	3	17	15	12	3	13	19	16	3	10	22	10	12
3P+0	5	125	115	113	2	155	149	145	4	158	194	108	86
	7	1030	810	800	10	1348	1036	1024	12	1496	1608	1078	530
	9	7472	5488	5472	16	9752	6824	6804	20	12658	13070	9942	3128
	11	51370	36438	36396	42	65476	43708	43660	48	102514	106606	87612	18994

FP ≙ Fehlende Punkte bei der Rekodierung
w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

Tabelle 17: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1
P-Q	3	13	19	16	3	17	15	12	3	10	22	10	12
0+3Q	5	155	149	145	4	125	115	113	2	158	194	108	86
	7	1348	1036	1024	12	1030	810	800	10	1496	1608	1078	530
	9	9752	6824	6804	20	7472	5488	5472	16	12658	13070	9942	3128
	11	65476	43708	43660	48	51370	36438	36396	42	102514	106606	87612	18994
P-Q	3	17	15	12	3	13	19	14	5	10	22	10	12
3P+Q	5	139	101	101	0	131	173	160	13	174	178	88	90
	7	1024	592	592	0	1515	1253	1198	55	1512	1336	822	514
	9	6308	3164	3164	0	12127	7921	7742	179	11430	9946	7346	2600
	11	34964	15660	15660	0	78881	47855	47189	666	87628	71508	59446	12062
P-Q	3	17	15	12	3	13	19	16	3	10	22	10	12
3P-Q	5	143	97	97	0	121	183	183	0	166	186	92	94
	7	1020	532	528	4	1361	1567	1543	24	1478	1498	846	652
	9	5710	2802	2802	0	12949	12123	12095	28	11816	12152	8052	4100
	11	30650	14182	14182	0	107507	86461	86303	158	94870	99562	75718	23844
P-Q	3	13	19	14	5	17	15	12	3	10	22	10	12
P+3Q	5	131	173	160	13	139	101	101	0	174	178	88	90
	7	1515	1253	1198	55	1024	592	592	0	1512	1336	822	514
	9	12127	7921	7742	179	6308	3164	3164	0	11430	9946	7346	2600
	11	78881	47855	47189	666	34964	15660	15660	0	87628	71508	59446	12062
P-Q	3	13	19	16	3	17	15	12	3	10	22	10	12
P-3Q	5	121	183	183	0	143	97	97	0	166	186	92	94
	7	1361	1567	1543	24	1020	532	528	4	1478	1498	846	652
	9	12949	12123	12095	28	5710	2802	2802	0	11816	12152	8052	4100
	11	107507	86461	86303	158	30650	14182	14182	0	94870	99562	75718	23844
P-Q	3	17	15	12	3	17	15	12	3	6	26	14	12
3P+3Q	5	139	101	101	0	139	101	101	0	134	282	196	86
	7	984	632	632	0	984	632	632	0	1806	2706	2216	490
	9	6048	4064	4064	0	6048	4064	4064	0	19230	24066	21112	2954
	11	38176	26848	26848	0	38176	26848	26848	0	175530	209526	188028	21498
3P+0	3	18	14	14	0	14	18	18	0	8	24	24	0
3P-3Q	5	124	100	100	0	152	136	136	0	136	248	248	0
	7	900	700	700	0	1204	972	972	0	1544	2424	2424	0
	9	6228	4972	4972	0	8636	6916	6916	0	16184	22600	22600	0
	11	43416	36136	36136	0	60688	49968	49968	0	157864	203736	203736	0
	13	308908	269268	269268	0	430284	369204	369204	0	1463912	1795864	1795864	0
3P+0	3	14	18	18	0	14	18	18	0	12	20	20	0
0+3Q	5	132	156	156	0	132	156	156	0	144	176	176	0
	7	1184	1312	1312	0	1184	1312	1312	0	1328	1488	1488	0
	9	10160	10832	10832	0	10160	10832	10832	0	11488	12320	12320	0
	11	84896	88416	88416	0	84896	88416	88416	0	96384	100736	100736	0
	13	698112	716544	716544	0	698112	716544	716544	0	794496	817280	817280	0
3P+0	3	18	14	14	0	10	22	22	0	12	20	20	0
3P+Q	5	124	100	100	0	146	206	206	0	168	152	152	0
	7	918	682	682	0	1516	1780	1780	0	1356	1076	1076	0
	9	6338	4574	4574	0	13782	14698	14698	0	9796	7420	7420	0
	11	42652	30532	30532	0	117274	117894	117894	0	68024	50696	50696	0
	13	284198	204314	204314	0	959504	926800	926800	0	464716	346420	346420	0
3P+0	3	18	14	14	0	10	22	22	0	12	20	20	0
3P-Q	5	124	100	100	0	146	206	206	0	168	152	152	0
	7	918	682	682	0	1516	1780	1780	0	1356	1076	1076	0
	9	6338	4574	4574	0	13782	14698	14698	0	9796	7420	7420	0
	11	42652	30532	30532	0	117274	117894	117894	0	68024	50696	50696	0
	13	284198	204314	204314	0	959504	926800	926800	0	464716	346420	346420	0
3P+0	3	14	18	18	0	14	18	18	0	12	20	20	0
P+3Q	5	122	166	166	0	152	136	136	0	168	152	152	0
	7	1272	1384	1380	4	1194	982	982	0	1324	1108	1108	0
	9	10888	11256	11224	32	8726	6986	6982	4	9716	8012	8004	8
	11	90020	90076	89780	296	62058	49718	49674	44	70140	58052	57980	72
3P+0	3	14	18	18	0	14	18	18	0	12	20	20	0
P-3Q	5	122	166	166	0	152	136	136	0	168	152	152	0
	7	1272	1384	1380	4	1194	982	982	0	1324	1108	1108	0
	9	10888	11256	11224	32	8726	6986	6982	4	9716	8012	8004	8
	11	90020	90076	89780	296	62058	49718	49674	44	70140	58052	57980	72

FP ≙ Fehlende Punkte bei der Rekodierung
w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

Tabelle 17: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1
3P+0	3	18	14	14	0	14	18	18	0	8	24	24	0
3P+3Q	5	124	100	100	0	152	136	136	0	136	248	248	0
	7	900	700	700	0	1204	972	972	0	1544	2424	2424	0
	9	6228	4972	4972	0	8636	6916	6916	0	16184	22600	22600	0
	11	43416	36136	36136	0	60688	49968	49968	0	157864	203736	203736	0
	13	308908	269268	269268	0	430284	369204	369204	0	1463912	1795864	1795864	0
0+3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P-3Q	5	152	136	136	0	124	100	100	0	136	248	248	0
	7	1204	972	972	0	900	700	700	0	1544	2424	2424	0
	9	8636	6916	6916	0	6228	4972	4972	0	16184	22600	22600	0
	11	60688	49968	49968	0	43416	36136	36136	0	157864	203736	203736	0
	13	430284	369204	369204	0	308908	269268	269268	0	1463912	1795864	1795864	0
0+3Q	3	14	18	18	0	14	18	18	0	12	20	20	0
3P+Q	5	152	136	136	0	122	166	166	0	168	152	152	0
	7	1194	982	982	0	1272	1384	1380	4	1324	1108	1108	0
	9	8726	6986	6982	4	10888	11256	11224	32	9716	8012	8004	8
	11	62058	49718	49674	44	90020	90076	89780	296	70140	58052	57980	72
0+3Q	3	14	18	18	0	14	18	18	0	12	20	20	0
3P-Q	5	152	136	136	0	122	166	166	0	168	152	152	0
	7	1194	982	982	0	1272	1384	1380	4	1324	1108	1108	0
	9	8726	6986	6982	4	10888	11256	11224	32	9716	8012	8004	8
	11	62058	49718	49674	44	90020	90076	89780	296	70140	58052	57980	72
0+3Q	3	10	22	22	0	18	14	14	0	12	20	20	0
P+3Q	5	146	206	206	0	124	100	100	0	168	152	152	0
	7	1516	1780	1780	0	918	682	682	0	1356	1076	1076	0
	9	13782	14698	14698	0	6338	4574	4574	0	9796	7420	7420	0
	11	117274	117894	117894	0	42652	30532	30532	0	68024	50696	50696	0
	13	959504	926800	926800	0	284198	204314	204314	0	464716	346420	346420	0
0+3Q	3	10	22	22	0	18	14	14	0	12	20	20	0
P-3Q	5	146	206	206	0	124	100	100	0	168	152	152	0
	7	1516	1780	1780	0	918	682	682	0	1356	1076	1076	0
	9	13782	14698	14698	0	6338	4574	4574	0	9796	7420	7420	0
	11	117274	117894	117894	0	42652	30532	30532	0	68024	50696	50696	0
	13	959504	926800	926800	0	284198	204314	204314	0	464716	346420	346420	0
0+3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P+3Q	5	152	136	136	0	124	100	100	0	136	248	248	0
	7	1204	972	972	0	900	700	700	0	1544	2424	2424	0
	9	8636	6916	6916	0	6228	4972	4972	0	16184	22600	22600	0
	11	60688	49968	49968	0	43416	36136	36136	0	157864	203736	203736	0
	13	430284	369204	369204	0	308908	269268	269268	0	1463912	1795864	1795864	0
3P+Q	3	18	14	14	0	14	18	18	0	8	24	24	0
3P-3Q	5	144	80	80	0	134	154	154	0	160	224	224	0
	7	846	434	434	0	1320	1144	1144	0	1632	1952	1952	0
	9	4498	2446	2446	0	10124	8180	8180	0	14880	16352	16352	0
	11	24774	14362	14362	0	70072	60808	60808	0	129984	131648	131648	0
	13	137426	92366	92366	0	516580	456348	456348	0	1072128	1034240	1034240	0
3P+Q	3	18	14	14	0	10	22	22	0	12	20	20	0
3P-Q	5	144	80	80	0	124	228	228	0	192	128	128	0
	7	860	420	420	0	1852	1796	1796	0	1336	712	712	0
	9	4600	2120	2120	0	16568	12168	12168	0	7664	3728	3728	0
	11	24128	9792	9792	0	115216	79472	79472	0	42112	17536	17536	0
	13	111552	45120	45120	0	781792	489760	489760	0	198528	82048	82048	0
3P+Q	3	14	18	18	0	14	18	18	0	12	20	20	0
P+3Q	5	134	154	154	0	134	154	154	0	176	144	144	0
	7	1304	1160	1160	0	1304	1160	1160	0	1256	1048	1048	0
	9	10188	8372	8348	24	10188	8372	8348	24	9584	7184	7184	0
	11	74824	59128	59032	96	74824	59128	59032	96	71056	43888	43888	0
3P+Q	3	14	18	18	0	14	18	18	0	12	20	20	0
P-3Q	5	134	154	154	0	134	154	154	0	192	128	128	0
	7	1348	1116	1116	0	1348	1116	1116	0	1336	712	712	0
	9	10480	7376	7376	0	10480	7376	7376	0	7632	3760	3760	0
	11	70344	47672	47672	0	70344	47672	47672	0	41696	18464	18464	0
	13	463240	299512	299512	0	463240	299512	299512	0	201600	93824	93824	0

FP ≙ Fehlende Punkte bei der Rekodierung
pos ≙ Anzahl der positiven Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

w ≙ Fensterbreite in Bits
neg ≙ Anzahl der negativen Rekodierungen
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

Tabelle 17: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1	pos	neg	pos-1	neg-1
3P+Q	3	18	14	14	0	14	18	18	0	8	24	24	0
3P+3Q	5	144	80	80	0	142	146	146	0	144	240	240	0
	7	846	434	434	0	1360	976	976	0	1592	2248	2248	0
	9	4474	2470	2470	0	9168	6448	6448	0	16784	19184	19184	0
	11	24738	14782	14782	0	57104	46064	46064	0	152504	154440	154440	0
	13	140118	96394	96394	0	399976	337048	337048	0	1255376	1215664	1215664	0
3P-Q	3	18	14	14	0	14	18	18	0	8	24	24	0
3P-3Q	5	144	80	80	0	142	146	146	0	144	240	240	0
	7	846	434	434	0	1360	976	976	0	1592	2248	2248	0
	9	4474	2470	2470	0	9168	6448	6448	0	16784	19184	19184	0
	11	24738	14782	14782	0	57104	46064	46064	0	152504	154440	154440	0
	13	140118	96394	96394	0	399976	337048	337048	0	1255376	1215664	1215664	0
3P-Q	3	14	18	18	0	14	18	18	0	12	20	20	0
P+3Q	5	134	154	154	0	134	154	154	0	192	128	128	0
	7	1348	1116	1116	0	1348	1116	1116	0	1336	712	712	0
	9	10480	7376	7376	0	10480	7376	7376	0	7632	3760	3760	0
	11	70344	47672	47672	0	70344	47672	47672	0	41696	18464	18464	0
	13	463240	299512	299512	0	463240	299512	299512	0	201600	93824	93824	0
3P-Q	3	14	18	18	0	14	18	18	0	12	20	20	0
P-3Q	5	134	154	154	0	134	154	154	0	176	144	144	0
	7	1304	1160	1160	0	1304	1160	1160	0	1256	1048	1048	0
	9	10188	8372	8348	24	10188	8372	8348	24	9584	7184	7184	0
	11	74824	59128	59032	96	74824	59128	59032	96	71056	43888	43888	0
3P-Q	3	18	14	14	0	14	18	18	0	8	24	24	0
3P+3Q	5	144	80	80	0	134	154	154	0	160	224	224	0
	7	846	434	434	0	1320	1144	1144	0	1632	1952	1952	0
	9	4498	2446	2446	0	10124	8180	8180	0	14880	16352	16352	0
	11	24774	14362	14362	0	70072	60808	60808	0	129984	131648	131648	0
	13	137426	92366	92366	0	516580	456348	456348	0	1072128	1034240	1034240	0
P+3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P-3Q	5	134	154	154	0	144	80	80	0	160	224	224	0
	7	1320	1144	1144	0	846	434	434	0	1632	1952	1952	0
	9	10124	8180	8180	0	4498	2446	2446	0	14880	16352	16352	0
	11	70072	60808	60808	0	24774	14362	14362	0	129984	131648	131648	0
	13	516580	456348	456348	0	137426	92366	92366	0	1072128	1034240	1034240	0
P+3Q	3	10	22	22	0	18	14	14	0	12	20	20	0
P-3Q	5	124	228	228	0	144	80	80	0	192	128	128	0
	7	1852	1796	1796	0	860	420	420	0	1336	712	712	0
	9	16568	12168	12168	0	4600	2120	2120	0	7664	3728	3728	0
	11	115216	79472	79472	0	24128	9792	9792	0	42112	17536	17536	0
	13	781792	489760	489760	0	111552	45120	45120	0	198528	82048	82048	0
P+3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P+3Q	5	142	146	146	0	144	80	80	0	144	240	240	0
	7	1360	976	976	0	846	434	434	0	1592	2248	2248	0
	9	9168	6448	6448	0	4474	2470	2470	0	16784	19184	19184	0
	11	57104	46064	46064	0	24738	14782	14782	0	152504	154440	154440	0
	13	399976	337048	337048	0	140118	96394	96394	0	1255376	1215664	1215664	0
P-3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P-3Q	5	142	146	146	0	144	80	80	0	144	240	240	0
	7	1360	976	976	0	846	434	434	0	1592	2248	2248	0
	9	9168	6448	6448	0	4474	2470	2470	0	16784	19184	19184	0
	11	57104	46064	46064	0	24738	14782	14782	0	152504	154440	154440	0
	13	399976	337048	337048	0	140118	96394	96394	0	1255376	1215664	1215664	0
P-3Q	3	14	18	18	0	18	14	14	0	8	24	24	0
3P+3Q	5	134	154	154	0	144	80	80	0	160	224	224	0
	7	1320	1144	1144	0	846	434	434	0	1632	1952	1952	0
	9	10124	8180	8180	0	4498	2446	2446	0	14880	16352	16352	0
	11	70072	60808	60808	0	24774	14362	14362	0	129984	131648	131648	0
	13	516580	456348	456348	0	137426	92366	92366	0	1072128	1034240	1034240	0
3P+3Q	3	18	14	14	0	18	14	14	0	4	28	28	0
3P-3Q	5	144	80	80	0	144	80	80	0	128	320	320	0
	7	832	448	448	0	832	448	448	0	1984	3136	3136	0
	9	4352	2816	2816	0	4352	2816	2816	0	21888	28288	28288	0
	11	25344	19712	19712	0	25344	19712	19712	0	207104	245504	245504	0
	13	165376	150016	150016	0	165376	150016	150016	0	1833472	2094592	2094592	0

FP ≙ Fehlende Punkte bei der Rekodierung
w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

A.1. Resultierende AJHD

Tabelle 18: Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten

FP	w	pos	neg	pos-1	neg-1	AJHD
3P+0	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	412	484	484	0	0,421875
	7	3648	4096	4096	0	0,4025380710659898
	9	31048	34488	34488	0	0,3940322628693994
	11	261968	289840	289840	0	0,38994456390755
	13	2203104	2434336	2434336	0	0,3878951085350366
3P+0	1	2	6	6	0	0,75
0+3Q	3	40	56	56	0	0,475
	5	408	488	488	0	0,4224537037037037
	7	3696	4112	4112	0	0,40276369168356996
	9	31808	33984	33984	0	0,3939240656986432
	11	266176	277568	277568	0	0,38967587251225844
	13	2190720	2250368	2250368	0	0,38757712187208243
3P+0	1	2	6	6	0	0,75
3P+Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3790	3538	3538	0	0,3974354879918242
	9	29916	26692	26692	0	0,38879001541670444
	11	227950	199122	199122	0	0,38496150546769325
	13	1708418	1477534	1477534	0	0,38322602681581397
3P+0	1	2	6	6	0	0,75
3P-Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3790	3538	3538	0	0,3974354879918242
	9	29916	26692	26692	0	0,38879001541670444
	11	227950	199122	199122	0	0,38496150546769325
	13	1708418	1477534	1477534	0	0,38322602681581397
3P+0	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	412	484	484	0	0,421875
	7	3648	4096	4096	0	0,4025380710659898
	9	31048	34488	34488	0	0,3940322628693994
	11	261968	289840	289840	0	0,38994456390755
	13	2203104	2434336	2434336	0	0,3878951085350366
0+3Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	412	484	484	0	0,421875
	7	3648	4096	4096	0	0,4025380710659898
	9	31048	34488	34488	0	0,3940322628693994
	11	261968	289840	289840	0	0,38994456390755
	13	2203104	2434336	2434336	0	0,3878951085350366
0+3Q	1	2	6	6	0	0,75
P+3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3790	3538	3538	0	0,3974354879918242
	9	29916	26692	26692	0	0,38879001541670444
	11	227950	199122	199122	0	0,38496150546769325
	13	1708418	1477534	1477534	0	0,38322602681581397
0+3Q	1	2	6	6	0	0,75
P-3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3790	3538	3538	0	0,3974354879918242
	9	29916	26692	26692	0	0,38879001541670444
	11	227950	199122	199122	0	0,38496150546769325
	13	1708418	1477534	1477534	0	0,38322602681581397
0+3Q	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	412	484	484	0	0,421875
	7	3648	4096	4096	0	0,4025380710659898
	9	31048	34488	34488	0	0,3940322628693994
	11	261968	289840	289840	0	0,38994456390755
	13	2203104	2434336	2434336	0	0,3878951085350366

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

Tabelle 18: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	AJHD
3P+Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3798	3530	3530	0	0,39737161471640264
	9	29502	26978	26978	0	0,38891166822867856
	11	224830	206818	206818	0	0,3851998195205271
	13	1726134	1582954	1582954	0	0,3834814211594587
3P+Q	1	2	6	6	0	0,75
3P-Q	3	40	56	56	0	0,475
	5	460	436	436	0	0,4149305555555556
	7	4048	2928	2928	0	0,3919578622816033
	9	28832	18016	18016	0	0,3829907975460123
	11	181456	106800	106800	0	0,37963383514760424
	13	1091872	616928	616928	0	0,37838921528508057
3P+Q	1	2	6	6	0	0,75
P-3Q	3	40	56	56	0	0,475
	5	460	436	436	0	0,4149305555555556
	7	4032	2944	2944	0	0,3920863309352518
	9	28592	18512	18512	0	0,38325717615309124
	11	182384	113808	113808	0	0,37992528965091227
	13	1128080	692848	692848	0	0,37866328065880883
3P+Q	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	430	466	466	0	0,4192708333333333
	7	3798	3658	3658	0	0,39860084140744517
	9	30426	28102	28102	0	0,3898688074085228
	11	234346	215286	215286	0	0,38599189070491136
	13	1795470	1649106	1649106	0	0,3842051644701724
3P-Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	430	466	466	0	0,4192708333333333
	7	3798	3658	3658	0	0,39860084140744517
	9	30426	28102	28102	0	0,3898688074085228
	11	234346	215286	215286	0	0,38599189070491136
	13	1795470	1649106	1649106	0	0,3842051644701724
3P-Q	1	2	6	6	0	0,75
P+3Q	3	40	56	56	0	0,475
	5	460	436	436	0	0,4149305555555556
	7	4032	2944	2944	0	0,3920863309352518
	9	28592	18512	18512	0	0,38325717615309124
	11	182384	113808	113808	0	0,37992528965091227
	13	1128080	692848	692848	0	0,37866328065880883
3P-Q	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3798	3530	3530	0	0,39737161471640264
	9	29502	26978	26978	0	0,38891166822867856
	11	224830	206818	206818	0	0,3851998195205271
	13	1726134	1582954	1582954	0	0,3834814211594587
P+3Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3798	3530	3530	0	0,39737161471640264
	9	29502	26978	26978	0	0,38891166822867856
	11	224830	206818	206818	0	0,3851998195205271
	13	1726134	1582954	1582954	0	0,3834814211594587
P+3Q	1	2	6	6	0	0,75
P-3Q	3	40	56	56	0	0,475
	5	460	436	436	0	0,4149305555555556
	7	4048	2928	2928	0	0,3919578622816033
	9	28832	18016	18016	0	0,3829907975460123
	11	181456	106800	106800	0	0,37963383514760424
	13	1091872	616928	616928	0	0,37838921528508057

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

Tabelle 18: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	AJHD
P+3Q	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	430	466	466	0	0,4192708333333333
	7	3798	3658	3658	0	0,39860084140744517
	9	30426	28102	28102	0	0,3898688074085228
	11	234346	215286	215286	0	0,38599189070491136
	13	1795470	1649106	1649106	0	0,3842051644701724
P-3Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	430	466	466	0	0,4192708333333333
	7	3798	3658	3658	0	0,39860084140744517
	9	30426	28102	28102	0	0,3898688074085228
	11	234346	215286	215286	0	0,38599189070491136
	13	1795470	1649106	1649106	0	0,3842051644701724
P-3Q	1	2	6	6	0	0,75
3P+3Q	3	40	56	56	0	0,475
	5	438	458	458	0	0,41811342592592593
	7	3798	3530	3530	0	0,39737161471640264
	9	29502	26978	26978	0	0,38891166822867856
	11	224830	206818	206818	0	0,3851998195205271
	13	1726134	1582954	1582954	0	0,3834814211594587
3P+3Q	1	2	6	6	0	0,75
3P-3Q	3	40	56	56	0	0,475
	5	416	480	480	0	0,4212962962962963
	7	3648	4032	4032	0	0,4019308943089431
	9	30592	33920	33920	0	0,3935649474689589
	11	257792	284928	284928	0	0,38954548084712415
	13	2164224	2394624	2394624	0	0,3875345239959568

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, neg ≙ Anzahl der negativen Rekodierungen
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1

A.2. Zusätzliche Nullspalten

Tabelle 19: Ergebnisse des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten und zusätzlichen Nullspalten

		Spalten beginnen mit $\begin{smallmatrix}1\\0\end{smallmatrix}$				Spalten beginnen mit $\begin{smallmatrix}0\\1\end{smallmatrix}$				Spalten beginnen mit $\begin{smallmatrix}1\\1\end{smallmatrix}$			
FP	w	pos	pos+1	neg	pos-1	pos	pos+1	neg	pos-1	pos	neg	pos-1	neg-1
3P+0	3	18	0	14	14	14	0	18	18	8	0	24	24
3P-3Q	5	124	0	100	100	152	0	136	136	136	0	248	248
	7	900	0	700	700	1204	0	972	972	1544	0	2424	2424
	9	6228	0	4972	4972	8636	0	6916	6916	16184	0	22600	22600
	11	43416	0	36136	36136	60688	0	49968	49968	157864	0	203736	203736
3P+0	3	14	0	18	18	14	0	18	18	12	0	20	20
0+3Q	5	132	0	156	156	132	0	156	156	144	0	176	176
	7	1184	0	1312	1312	1184	0	1312	1312	1328	0	1488	1488
	9	10160	0	10832	10832	10160	0	10832	10832	11488	0	12320	12320
	11	84896	0	88416	88416	84896	0	88416	88416	96384	0	100736	100736
3P+0	3	18	0	14	14	10	0	22	22	12	0	20	20
3P+Q	5	124	0	100	100	146	0	206	206	168	0	152	152
	7	918	0	682	682	1516	0	1780	1780	1356	0	1076	1076
	9	6338	0	4574	4574	13782	0	14698	14698	9796	0	7420	7420
	11	42652	0	30532	30532	117274	0	117894	117894	68024	0	50696	50696
3P+0	3	18	0	14	14	10	0	22	22	12	0	20	20
3P-Q	5	124	0	100	100	146	0	206	206	168	0	152	152
	7	918	0	682	682	1516	0	1780	1780	1356	0	1076	1076
	9	6338	0	4574	4574	13782	0	14698	14698	9796	0	7420	7420
	11	42652	0	30532	30532	117274	0	117894	117894	68024	0	50696	50696
3P+0	3	18	0	14	14	14	0	18	18	8	0	24	24
3P+3Q	5	124	0	100	100	152	0	136	136	136	0	248	248
	7	900	0	700	700	1204	0	972	972	1544	0	2424	2424
	9	6228	0	4972	4972	8636	0	6916	6916	16184	0	22600	22600
	11	43416	0	36136	36136	60688	0	49968	49968	157864	0	203736	203736
0+3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P-3Q	5	152	0	136	136	124	0	100	100	136	0	248	248
	7	1204	0	972	972	900	0	700	700	1544	0	2424	2424
	9	8636	0	6916	6916	6228	0	4972	4972	16184	0	22600	22600
	11	60688	0	49968	49968	43416	0	36136	36136	157864	0	203736	203736
0+3Q	3	10	0	22	22	18	0	14	14	12	0	20	20
P+3Q	5	146	0	206	206	124	0	100	100	168	0	152	152
	7	1516	0	1780	1780	918	0	682	682	1356	0	1076	1076
	9	13782	0	14698	14698	6338	0	4574	4574	9796	0	7420	7420
	11	117274	0	117894	117894	42652	0	30532	30532	68024	0	50696	50696
0+3Q	3	10	0	22	22	18	0	14	14	12	0	20	20
P-3Q	5	146	0	206	206	124	0	100	100	168	0	152	152
	7	1516	0	1780	1780	918	0	682	682	1356	0	1076	1076
	9	13782	0	14698	14698	6338	0	4574	4574	9796	0	7420	7420
	11	117274	0	117894	117894	42652	0	30532	30532	68024	0	50696	50696
0+3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P+3Q	5	152	0	136	136	124	0	100	100	136	0	248	248
	7	1204	0	972	972	900	0	700	700	1544	0	2424	2424
	9	8636	0	6916	6916	6228	0	4972	4972	16184	0	22600	22600
	11	60688	0	49968	49968	43416	0	36136	36136	157864	0	203736	203736
3P+Q	3	18	0	14	14	14	0	18	18	8	0	24	24
3P-3Q	5	144	0	80	80	134	0	154	154	160	0	224	224
	7	846	0	434	434	1320	0	1144	1144	1632	0	1952	1952
	9	4498	0	2446	2446	10051	73	8180	8180	14880	0	16352	16352
	11	24701	73	14362	14362	70052	20	60808	60808	129984	0	131648	131648
3P+Q	3	18	0	14	14	10	0	22	22	12	0	20	20
3P-Q	5	144	0	80	80	124	0	228	228	192	0	128	128
	7	860	0	420	420	1852	0	1796	1796	1336	0	712	712
	9	4600	0	2120	2120	16412	156	12168	12168	7664	0	3728	3728
	11	23972	156	9792	9792	115012	204	79472	79472	41800	312	17536	17536
3P+Q	3	14	0	18	18	14	0	18	18	12	0	20	20
P-3Q	5	134	0	154	154	134	0	154	154	192	0	128	128
	7	1348	0	1116	1116	1348	0	1116	1116	1336	0	712	712
	9	10422	58	7376	7376	10422	58	7376	7376	7632	0	3760	3760
	11	70192	152	47672	47672	70192	152	47672	47672	41464	232	18464	18464

FP ≙ Fehlende Punkte bei der Rekodierung — w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen — neg ≙ Anzahl der negativen Rekodierungen
pos+1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten+1 — pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

Tabelle 19: (Fortsetzung)

FP	w	pos	pos+1	neg	pos-1	pos	pos+1	neg	pos-1	pos	pos+1	neg	pos-1
3P+Q	3	18	0	14	14	14	0	18	18	8	0	24	24
3P+3Q	5	144	0	80	80	142	0	146	146	144	0	240	240
	7	846	0	434	434	1360	0	976	976	1592	0	2248	2248
	9	4474	0	2470	2470	9095	73	6448	6448	16784	0	19184	19184
	11	24665	73	14782	14782	57084	20	46064	46064	152212	292	154440	154440
3P-Q	3	18	0	14	14	14	0	18	18	8	0	24	24
3P-3Q	5	144	0	80	80	142	0	146	146	144	0	240	240
	7	846	0	434	434	1360	0	976	976	1592	0	2248	2248
	9	4474	0	2470	2470	9095	73	6448	6448	16784	0	19184	19184
	11	24665	73	14782	14782	57084	20	46064	46064	152212	292	154440	154440
3P-Q	3	14	0	18	18	14	0	18	18	12	0	20	20
P+3Q	5	134	0	154	154	134	0	154	154	192	0	128	128
	7	1348	0	1116	1116	1348	0	1116	1116	1336	0	712	712
	9	10422	58	7376	7376	10422	58	7376	7376	7632	0	3760	3760
	11	70192	152	47672	47672	70192	152	47672	47672	41464	232	18464	18464
3P-Q	3	18	0	14	14	14	0	18	18	8	0	24	24
3P+3Q	5	144	0	80	80	134	0	154	154	160	0	224	224
	7	846	0	434	434	1320	0	1144	1144	1632	0	1952	1952
	9	4498	0	2446	2446	10051	73	8180	8180	14880	0	16352	16352
	11	24701	73	14362	14362	70052	20	60808	60808	129984	0	131648	131648
P+3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P-3Q	5	134	0	154	154	144	0	80	80	160	0	224	224
	7	1320	0	1144	1144	846	0	434	434	1632	0	1952	1952
	9	10051	73	8180	8180	4498	0	2446	2446	14880	0	16352	16352
	11	70052	20	60808	60808	24701	73	14362	14362	129984	0	131648	131648
P+3Q	3	10	0	22	22	18	0	14	14	12	0	20	20
P-3Q	5	124	0	228	228	144	0	80	80	192	0	128	128
	7	1852	0	1796	1796	860	0	420	420	1336	0	712	712
	9	16412	156	12168	12168	4600	0	2120	2120	7664	0	3728	3728
	11	115012	204	79472	79472	23972	156	9792	9792	41800	312	17536	17536
P+3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P+3Q	5	142	0	146	146	144	0	80	80	144	0	240	240
	7	1360	0	976	976	846	0	434	434	1592	0	2248	2248
	9	9095	73	6448	6448	4474	0	2470	2470	16784	0	19184	19184
	11	57084	20	46064	46064	24665	73	14782	14782	152212	292	154440	154440
P-3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P-3Q	5	142	0	146	146	144	0	80	80	144	0	240	240
	7	1360	0	976	976	846	0	434	434	1592	0	2248	2248
	9	9095	73	6448	6448	4474	0	2470	2470	16784	0	19184	19184
	11	57084	20	46064	46064	24665	73	14782	14782	152212	292	154440	154440
P-3Q	3	14	0	18	18	18	0	14	14	8	0	24	24
3P+3Q	5	134	0	154	154	144	0	80	80	160	0	224	224
	7	1320	0	1144	1144	846	0	434	434	1632	0	1952	1952
	9	10051	73	8180	8180	4498	0	2446	2446	14880	0	16352	16352
	11	70052	20	60808	60808	24701	73	14362	14362	129984	0	131648	131648
3P+3Q	3	18	0	14	14	18	0	14	14	4	0	28	28
3P-3Q	5	144	0	80	80	144	0	80	80	128	0	320	320
	7	832	0	448	448	832	0	448	448	1984	0	3136	3136
	9	4352	0	2816	2816	4352	0	2816	2816	21888	0	28288	28288
	11	25344	0	19712	19712	25344	0	19712	19712	207104	0	245504	245504

FP ≙ Fehlende Punkte bei der Rekodierung
w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen
pos+1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten+1
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

A.3. Resultierende AJHD mit zusätzlichen Nullspalten

Tabelle 20: Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit acht vorzuberechnenden Punkten und zusätzlichen Nullspalten

FP	w	pos	pos+1	neg	pos-1	AJHD
3P+Q	1	2	0	6	6	0,75
3P-3Q	3	40	0	56	56	0,475
	5	438	0	458	458	0,41811342592592593
	7	3798	0	3530	3530	0,39737161471640264
	9	29429	73	26978	26978	0,3888771842972458
	11	224737	93	206818	206818	0,3851635151855189
3P+Q	1	2	0	6	6	0,75
3P-Q	3	40	0	56	56	0,475
	5	460	0	436	436	0,4149305555555556
	7	4048	0	2928	2928	0,3919578622816033
	9	28676	156	18016	18016	0,382916027607362
	11	180784	672	106800	106800	0,3795405459397521
3P+Q	1	2	0	6	6	0,75
P-3Q	3	40	0	56	56	0,475
	5	460	0	436	436	0,4149305555555556
	7	4032	0	2944	2944	0,3920863309352518
	9	28476	116	18512	18512	0,38320159163395484
	11	181848	536	113808	113808	0,3798549014659575
3P+Q	1	2	0	6	6	0,75
3P+3Q	3	40	0	56	56	0,475
	5	430	0	466	466	0,4192708333333333
	7	3798	0	3658	3658	0,39860084140744517
	9	30353	73	28102	28102	0,3898344564071676
	11	233961	385	215286	215286	0,3859473936471611
3P-Q	1	2	0	6	6	0,75
3P-3Q	3	40	0	56	56	0,475
	5	430	0	466	466	0,4192708333333333
	7	3798	0	3658	3658	0,39860084140744517
	9	30353	73	28102	28102	0,3898344564071676
	11	233961	385	215286	215286	0,3859473936471611
3P-Q	1	2	0	6	6	0,75
P+3Q	3	40	0	56	56	0,475
	5	460	0	436	436	0,4149305555555556
	7	4032	0	2944	2944	0,3920863309352518
	9	28476	116	18512	18512	0,38320159163395484
	11	181848	536	113808	113808	0,3798549014659575
3P-Q	1	2	0	6	6	0,75
3P+3Q	3	40	0	56	56	0,475
	5	438	0	458	458	0,41811342592592593
	7	3798	0	3530	3530	0,39737161471640264
	9	29429	73	26978	26978	0,3888771842972458
	11	224737	93	206818	206818	0,3851635151855189
P+3Q	1	2	0	6	6	0,75
3P-3Q	3	40	0	56	56	0,475
	5	438	0	458	458	0,41811342592592593
	7	3798	0	3530	3530	0,39737161471640264
	9	29429	73	26978	26978	0,3888771842972458
	11	224737	93	206818	206818	0,3851635151855189
P+3Q	1	2	0	6	6	0,75
P-3Q	3	40	0	56	56	0,475
	5	460	0	436	436	0,4149305555555556
	7	4048	0	2928	2928	0,3919578622816033
	9	28676	156	18016	18016	0,382916027607362
	11	180784	672	106800	106800	0,3795405459397521
P+3Q	1	2	0	6	6	0,75
3P+3Q	3	40	0	56	56	0,475
	5	430	0	466	466	0,4192708333333333
	7	3798	0	3658	3658	0,39860084140744517
	9	30353	73	28102	28102	0,3898344564071676
	11	233961	385	215286	215286	0,3859473936471611

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, neg ≙ Anzahl der negativen Rekodierungen
pos+1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten+1
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

Tabelle 20: (Fortsetzung)

FP	w	pos	pos+1	neg	pos-1	AJHD
P-3Q	1	2	0	6	6	0,75
3P-3Q	3	40	0	56	56	0,475
	5	430	0	466	466	0,4192708333333333
	7	3798	0	3658	3658	0,39860084140744517
	9	30353	73	28102	28102	0,3898344564071676
	11	233961	385	215286	215286	0,3859473936471611
P-3Q	1	2	0	6	6	0,75
3P+3Q	3	40	0	56	56	0,475
	5	438	0	458	458	0,41811342592592593
	7	3798	0	3530	3530	0,39737161471640264
	9	29429	73	26978	26978	0,3888771842972458
	11	224737	93	206818	206818	0,3851635151855189

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits
pos ≙ Anzahl der positiven Rekodierungen, neg ≙ Anzahl der negativen Rekodierungen
pos+1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten+1
pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1

B. Tabellen für die FW-Sh Methode mit vier vorzuberechnenden Punkten

Tabelle 21: Ergebnisse des FW-Shamir Verfahrens mit vier vorzuberechnenden Punkten

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	235	917	308	609	24	585	∼ 0,646417025862069
0+3Q	7	2238	12434	3437	8997	396	8601	∼ 0,650201177360121
3P+Q	9	22083	176861	38162	138699	5560	133139	∼ 0,6635995196420726
3P-3Q	11	225348	2604428	428640	2175788	76684	2099104	∼ 0,6798388868992055
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	258	894	350	544	34	510	∼ 0,6244612068965517
0+3Q	7	2750	11554	4454	7100	744	6356	∼ 0,601647307859314
3P+Q	9	30180	154684	56436	98248	11192	87056	∼ 0,5923592375366569
3P-Q	11	342110	2132834	714150	1418684	165892	1252792	∼ 0,5898883095495837
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	244	908	344	564	32	532	∼ 0,6320043103448276
0+3Q	7	2612	11916	4346	7570	596	6974	∼ 0,6153273809523809
3P+Q	9	28628	162028	53942	108086	9432	98654	∼ 0,6106514166550165
P+3Q	11	326844	2265604	676474	1589130	145844	1443286	∼ 0,6105892569642908
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	250	902	340	562	34	528	∼ 0,6303879310344828
0+3Q	7	2584	11848	4280	7568	748	6820	∼ 0,612917208495882
3P+Q	9	27990	161578	54166	107412	11640	95772	∼ 0,6072994969184797
P-3Q	11	317072	2268176	689160	1579016	178924	1400092	∼ 0,6071337440368589
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	236	916	310	606	24	582	∼ 0,6454741379310344
0+3Q	7	2233	12423	3459	8964	379	8585	∼ 0,6497204515989627
3P+Q	9	22176	176592	38775	137817	5155	132662	∼ 0,6627174862092493
3P+3Q	11	227265	2598207	438084	2160123	72613	2087510	∼ 0,6785523767921606
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	236	916	310	606	24	582	∼ 0,6454741379310344
0+3Q	7	2233	12423	3459	8964	379	8585	∼ 0,6497204515989627
3P-Q	9	22176	176592	38775	137817	5155	132662	∼ 0,6627174862092493
3P-3Q	11	227265	2598207	438084	2160123	72613	2087510	∼ 0,6785523767921606
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	250	902	340	562	34	528	∼ 0,6303879310344828
0+3Q	7	2584	11848	4280	7568	748	6820	∼ 0,612917208495882
3P-Q	9	27990	161578	54166	107412	11640	95772	∼ 0,6072994969184797
P+3Q	11	317072	2268176	689160	1579016	178924	1400092	∼ 0,6071337440368589
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	244	908	344	564	32	532	∼ 0,6320043103448276
0+3Q	7	2612	11916	4346	7570	596	6974	∼ 0,6153273809523809
3P-Q	9	28628	162028	53942	108086	9432	98654	∼ 0,6106514166550165
P-3Q	11	326844	2265604	676474	1589130	145844	1443286	∼ 0,6105892569642908
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	235	917	308	609	24	585	∼ 0,646417025862069
0+3Q	7	2238	12434	3437	8997	396	8601	∼ 0,650201177360121
3P-Q	9	22083	176861	38162	138699	5560	133139	∼ 0,6635995196420726
3P+3Q	11	225348	2604428	428640	2175788	76684	2099104	∼ 0,6798388868992055
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	235	917	308	609	24	585	∼ 0,646417025862069
0+3Q	7	2238	12434	3437	8997	396	8601	∼ 0,650201177360121
P+3Q	9	22083	176861	38162	138699	5560	133139	∼ 0,6635995196420726
3P-3Q	11	225348	2604428	428640	2175788	76684	2099104	∼ 0,6798388868992055

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	258	894	350	544	34	510	∼ 0,6244612068965517
0+3Q	7	2750	11554	4454	7100	744	6356	∼ 0,601647307859314
P+3Q	9	30180	154684	56436	98248	11192	87056	∼ 0,5923592375366569
P-3Q	11	342110	2132834	714150	1418684	165892	1252792	∼ 0,5898883095495837
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	236	916	310	606	24	582	∼ 0,6454741379310344
0+3Q	7	2233	12423	3459	8964	379	8585	∼ 0,6497204515989627
P+3Q	9	22176	176592	38775	137817	5155	132662	∼ 0,6627174862092493
3P+3Q	11	227265	2598207	438084	2160123	72613	2087510	∼ 0,6785523767921606
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	236	916	310	606	24	582	∼ 0,6454741379310344
0+3Q	7	2233	12423	3459	8964	379	8585	∼ 0,6497204515989627
P-3Q	9	22176	176592	38775	137817	5155	132662	∼ 0,6627174862092493
3P-3Q	11	227265	2598207	438084	2160123	72613	2087510	∼ 0,6785523767921606
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	235	917	308	609	24	585	∼ 0,646417025862069
0+3Q	7	2238	12434	3437	8997	396	8601	∼ 0,650201177360121
P-3Q	9	22083	176861	38162	138699	5560	133139	∼ 0,6635995196420726
3P+3Q	11	225348	2604428	428640	2175788	76684	2099104	∼ 0,6798388868992055
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	24	48	4	44	∼ 0,675
3P+0	5	220	932	276	656	16	640	∼ 0,662176724137931
0+3Q	7	1928	12984	2700	10284	168	10116	∼ 0,6824666236003445
3P+3Q	9	17720	190024	26700	163324	2176	161148	∼ 0,7057298826871964
3P-3Q	11	166340	2874044	267520	2606524	29044	2577480	∼ 0,7283962191610909
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	195	957	232	725	26	699	∼ 0,6827855603448275
3P+Q	7	1757	13555	2518	11037	403	10634	∼ 0,6972852859284644
3P-Q	9	16546	200334	27907	172427	5072	167355	∼ 0,7169332589311005
3P-3Q	11	163819	3041525	313709	2727816	65239	2662577	∼ 0,7369285232108593
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	220	932	263	669	34	635	∼ 0,6632543103448276
3P+Q	7	2247	12665	3167	9498	610	8888	∼ 0,6585042527993109
3P-Q	9	23594	179046	38829	140217	8593	131624	∼ 0,6624109596522899
P+3Q	11	259827	2604909	478981	2125928	122773	2003155	∼ 0,6701066704339245
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	220	932	263	669	34	635	∼ 0,6632543103448276
3P+Q	7	2247	12665	3167	9498	610	8888	∼ 0,6585042527993109
3P-Q	9	23594	179046	38829	140217	8593	131624	∼ 0,6624109596522899
P-3Q	11	259827	2604909	478981	2125928	122773	2003155	∼ 0,6701066704339245
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	195	957	232	725	26	699	∼ 0,6827855603448275
3P+Q	7	1757	13555	2518	11037	403	10634	∼ 0,6972852859284644
3P-Q	9	16546	200334	27907	172427	5072	167355	∼ 0,7169332589311005
3P+3Q	11	163819	3041525	313709	2727816	65239	2662577	∼ 0,7369285232108593
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	193	959	275	684	18	666	∼ 0,6730872844827587
3P+Q	7	1871	13473	3021	10452	399	10053	∼ 0,6849644080496682
P+3	9	18450	197118	34380	162738	5969	156769	∼ 0,7012333633756193
Q3P-3Q	11	192405	2961483	395359	2566124	88757	2477367	∼ 0,7184301486451032
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	226	926	294	632	27	605	∼ 0,6534213362068966
3P+Q	7	2484	12332	3710	8622	584	8038	∼ 0,6391089909443726
P+3Q	9	27093	170219	47504	122715	9347	113368	∼ 0,6339315634388722
P-3Q	11	312077	2411427	606401	1805026	145197	1659829	∼ 0,6338840250144868

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	203	949	253	696	18	678	∼ 0,6749730603448276
3P+Q	7	1892	13292	2743	10549	269	10280	∼ 0,6878419330615748
P+3Q	9	18089	194583	30245	164338	3827	160511	∼ 0,7056742784631492
3P+3Q	11	181649	2931679	335352	2596327	55645	2540682	∼ 0,7244735690306167
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	200	952	273	679	19	660	∼ 0,6706627155172413
3P+Q	7	1949	13283	3030	10253	487	9766	∼ 0,6787896655231561
P-3Q	9	18922	193606	35507	158099	7210	150889	∼ 0,6927562651438453
3P-3Q	11	196386	2901310	418002	2483308	107581	2375727	∼ 0,7088365904889514
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	205	947	255	692	19	673	∼ 0,6734913793103449
3P+Q	7	1947	13205	2826	10379	401	9978	∼ 0,6821474565357373
P-3Q	9	18724	192556	32232	160324	5780	154544	∼ 0,6974267513020092
3P+3Q	11	190996	2889900	371266	2518634	82734	2435900	∼ 0,7144839387049494
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	183	969	228	741	12	729	∼ 0,6905980603448276
3P+Q	7	1570	13934	2144	11790	198	11592	∼ 0,7171277504806666
3P+3Q	9	13735	209209	21504	187705	2524	185181	∼ 0,7428490515591579
3P-3Q	11	126469	3220875	222029	2998846	33113	2965733	∼ 0,7661845043112833
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	205	947	255	692	19	673	∼ 0,6734913793103449
3P-Q	7	1947	13205	2826	10379	401	9978	∼ 0,6821474565357373
P+3Q	9	18724	192556	32232	160324	5780	154544	∼ 0,6974267513020092
3P-3Q	11	190996	2889900	371266	2518634	82734	2435900	∼ 0,7144839387049494
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	226	926	294	632	27	605	∼ 0,6534213362068966
3P-Q	7	2484	12332	3710	8622	584	8038	∼ 0,6391089909443726
P+3Q	9	27093	170219	47504	122715	9347	113368	∼ 0,6339315634388722
P-3Q	11	312077	2411427	606401	1805026	145197	1659829	∼ 0,6338840250144868
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	200	952	273	679	19	660	∼ 0,6706627155172413
3P-Q	7	1949	13283	3030	10253	487	9766	∼ 0,6787896655231561
P+3Q	9	18922	193606	35507	158099	7210	150889	∼ 0,6927562651438453
3P+3Q	11	196386	2901310	418002	2483308	107581	2375727	∼ 0,7088365904889514
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	203	949	253	696	18	678	∼ 0,6749730603448276
3P-Q	7	1892	13292	2743	10549	269	10280	∼ 0,6878419330615748
P-3Q	9	18089	194583	30245	164338	3827	160511	∼ 0,7056742784631492
3P-3Q	11	181649	2931679	335352	2596327	55645	2540682	∼ 0,7244735690306167
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	193	959	275	684	18	666	∼ 0,6730872844827587
3P-Q	7	1871	13473	3021	10452	399	10053	∼ 0,6849644080496682
P-3Q	9	18450	197118	34380	162738	5969	156769	∼ 0,7012333633756193
3P+3Q	11	192405	2961483	395359	2566124	88757	2477367	∼ 0,7184301486451032
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	183	969	228	741	12	729	∼ 0,6905980603448276
3P-Q	7	1570	13934	2144	11790	198	11592	∼ 0,7171277504806666
3P+3Q	9	13735	209209	21504	187705	2524	185181	∼ 0,7428490515591579
3P-3Q	11	126469	3220875	222029	2998846	33113	2965733	∼ 0,7661845043112833
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	219	933	274	659	18	641	∼ 0,6628502155172413
P+3Q	7	2292	12636	3337	9299	364	8935	∼ 0,6575686221743812
P-3Q	9	23297	178879	40500	138379	6497	131882	∼ 0,6619928469570817
3P-3Q	11	251454	2610610	491074	2119536	102926	2016610	∼ 0,6712046825990734

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	219	933	274	659	18	641	∼ 0,6628502155172413
P+3Q	7	2292	12636	3337	9299	364	8935	∼ 0,6575686221743812
P-3Q	9	23297	178879	40500	138379	6497	131882	∼ 0,6619928469570817
3P+3Q	11	251454	2610610	491074	2119536	102926	2016610	∼ 0,6712046825990734
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	198	954	243	711	10	701	∼ 0,6807650862068966
P+3Q	7	1830	13434	2522	10912	214	10698	∼ 0,6966218388341192
3P+3Q	9	16827	198117	27333	170784	3599	167185	∼ 0,7157991826541036
3P-3Q	11	162936	3006936	299903	2707033	55570	2651463	∼ 0,7357132009938061
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
3P+0	5	198	954	243	711	10	701	∼ 0,6807650862068966
P-3Q	7	1830	13434	2522	10912	214	10698	∼ 0,6966218388341192
3P+3Q	9	16827	198117	27333	170784	3599	167185	∼ 0,7157991826541036
3P-3Q	11	162936	3006936	299903	2707033	55570	2651463	∼ 0,7357132009938061
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	219	933	274	659	18	641	∼ 0,6628502155172413
3P+Q	7	2292	12636	3337	9299	364	8935	∼ 0,6575686221743812
3P-Q	9	23297	178879	40500	138379	6497	131882	∼ 0,6619928469570817
3P-3Q	11	251454	2610610	491074	2119536	102926	2016610	∼ 0,6712046825990734
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	226	926	294	632	27	605	∼ 0,6534213362068966
3P+Q	7	2484	12332	3710	8622	584	8038	∼ 0,6391089909443726
3P-Q	9	27093	170219	47504	122715	9347	113368	∼ 0,6339315634388722
P+3Q	11	312077	2411427	606401	1805026	145197	1659829	∼ 0,6338840250144868
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	226	926	294	632	27	605	∼ 0,6534213362068966
3P+Q	7	2484	12332	3710	8622	584	8038	∼ 0,6391089909443726
3P-Q	9	27093	170219	47504	122715	9347	113368	∼ 0,6339315634388722
P-3Q	11	312077	2411427	606401	1805026	145197	1659829	∼ 0,6338840250144868
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	219	933	274	659	18	641	∼ 0,6628502155172413
3P+Q	7	2292	12636	3337	9299	364	8935	∼ 0,6575686221743812
3P-Q	9	23297	178879	40500	138379	6497	131882	∼ 0,6619928469570817
3P+3Q	11	251454	2610610	491074	2119536	102926	2016610	∼ 0,6712046825990734
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	193	959	275	684	18	666	∼ 0,6730872844827587
3P+Q	7	1871	13473	3021	10452	399	10053	∼ 0,6849644080496682
P+3Q	9	18450	197118	34380	162738	5969	156769	∼ 0,7012333633756193
3P-3Q	11	192405	2961483	395359	2566124	88757	2477367	∼ 0,7184301486451032
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	220	932	263	669	34	635	∼ 0,6632543103448276
3P+Q	7	2247	12665	3167	9498	610	8888	∼ 0,6585042527993109
P+3Q	9	23594	179046	38829	140217	8593	131624	∼ 0,6624109596522899
P-3Q	11	259827	2604909	478981	2125928	122773	2003155	∼ 0,6701066704339245
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	203	949	253	696	18	678	∼ 0,6749730603448276
3P+Q	7	1892	13292	2743	10549	269	10280	∼ 0,6878419330615748
P+3Q	9	18089	194583	30245	164338	3827	160511	∼ 0,7056742784631492
3P+3Q	11	181649	2931679	335352	2596327	55645	2540682	∼ 0,7244735690306167
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	205	947	255	692	19	673	∼ 0,6734913793103449
3P+Q	7	1947	13205	2826	10379	401	9978	∼ 0,6821474565357373
P-3Q	9	18724	192556	32232	160324	5780	154544	∼ 0,6974267513020092
3P-3Q	11	190996	2889900	371266	2518634	82734	2435900	∼ 0,7144839387049494

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	200	952	273	679	19	660	∼ 0,6706627155172413
3P+Q	7	1949	13283	3030	10253	487	9766	∼ 0,6787896655231561
P-3Q	9	18922	193606	35507	158099	7210	150889	∼ 0,6927562651438453
3P+3Q	11	196386	2901310	418002	2483308	107581	2375727	∼ 0,7088365904889514
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	198	954	243	711	10	701	∼ 0,6807650862068966
3P+Q	7	1830	13434	2522	10912	214	10698	∼ 0,6966218388341192
3P+3Q	9	16827	198117	27333	170784	3599	167185	∼ 0,7157991826541036
3P-3Q	11	162936	3006936	299903	2707033	55570	2651463	∼ 0,7357132009938061
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	200	952	273	679	19	660	∼ 0,6706627155172413
3P-Q	7	1949	13283	3030	10253	487	9766	∼ 0,6787896655231561
P+3Q	9	18922	193606	35507	158099	7210	150889	∼ 0,6927562651438453
3P-3Q	11	196386	2901310	418002	2483308	107581	2375727	∼ 0,7088365904889514
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	220	932	263	669	34	635	∼ 0,6632543103448276
3P-Q	7	2247	12665	3167	9498	610	8888	∼ 0,6585042527993109
P+3Q	9	23594	179046	38829	140217	8593	131624	∼ 0,6624109596522899
P-3Q	11	259827	2604909	478981	2125928	122773	2003155	∼ 0,6701066704339245
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	205	947	255	692	19	673	∼ 0,6734913793103449
3P-Q	7	1947	13205	2826	10379	401	9978	∼ 0,6821474565357373
P+3Q	9	18724	192556	32232	160324	5780	154544	∼ 0,6974267513020092
3P+3Q	11	190996	2889900	371266	2518634	82734	2435900	∼ 0,7144839387049494
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	203	949	253	696	18	678	∼ 0,6749730603448276
3P-Q	7	1892	13292	2743	10549	269	10280	∼ 0,6878419330615748
P-3Q	9	18089	194583	30245	164338	3827	160511	∼ 0,7056742784631492
3P-3Q	11	181649	2931679	335352	2596327	55645	2540682	∼ 0,7244735690306167
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	193	959	275	684	18	666	∼ 0,6730872844827587
3P-Q	7	1871	13473	3021	10452	399	10053	∼ 0,6849644080496682
P-3Q	9	18450	197118	34380	162738	5969	156769	∼ 0,7012333633756193
3P+3Q	11	192405	2961483	395359	2566124	88757	2477367	∼ 0,7184301486451032
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	198	954	243	711	10	701	∼ 0,6807650862068966
3P-Q	7	1830	13434	2522	10912	214	10698	∼ 0,6966218388341192
3P+3Q	9	16827	198117	27333	170784	3599	167185	∼ 0,7157991826541036
3P-3Q	11	162936	3006936	299903	2707033	55570	2651463	∼ 0,7357132009938061
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	195	957	232	725	26	699	∼ 0,6827855603448275
P+3Q	7	1757	13555	2518	11037	403	10634	∼ 0,6972852859284644
P-3Q	9	16546	200334	27907	172427	5072	167355	∼ 0,7169332589311005
3P-3Q	11	163819	3041525	313709	2727816	65239	2662577	∼ 0,7369285232108593
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	195	957	232	725	26	699	∼ 0,6827855603448275
P+3Q	7	1757	13555	2518	11037	403	10634	∼ 0,6972852859284644
P-3Q	9	16546	200334	27907	172427	5072	167355	∼ 0,7169332589311005
3P+3Q	11	163819	3041525	313709	2727816	65239	2662577	∼ 0,7369285232108593
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	183	969	228	741	12	729	∼ 0,6905980603448276
P+3Q	7	1570	13934	2144	11790	198	11592	∼ 0,7171277504806666
3P+3Q	9	13735	209209	21504	187705	2524	185181	∼ 0,7428490515591579
3P-3Q	11	126469	3220875	222029	2998846	33113	2965733	∼ 0,7661845043112833

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	20	52	4	48	∼ 0,6874999999999999
0+3Q	5	183	969	228	741	12	729	∼ 0,6905980603448276
P-3Q	7	1570	13934	2144	11790	198	11592	∼ 0,7171277504806666
3P+3Q	9	13735	209209	21504	187705	2524	185181	∼ 0,7428490515591579
3P-3Q	11	126469	3220875	222029	2998846	33113	2965733	∼ 0,7661845043112833
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	164	988	245	743	19	724	∼ 0,6927532327586207
3P-Q	7	1775	14033	3212	10821	719	10102	∼ 0,6906515957446808
P+3Q	9	21292	203236	43969	159267	15311	143956	∼ 0,6863894943574687
3P-3Q	11	264115	2987661	602030	2385631	279808	2105823	∼ 0,6839839812528051
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	208	944	208	736	32	704	∼ 0,6831896551724138
3P-Q	7	2344	12760	3072	9688	412	9276	∼ 0,6652169243986255
P+3Q	9	28340	175820	46196	129624	10264	119360	∼ 0,6437508595397451
P-3Q	11	376760	2436360	680144	1756216	208272	1547944	∼ 0,6230593746739649
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	181	971	211	760	19	741	∼ 0,6950431034482759
3P-Q	7	1905	13631	2939	10692	474	10218	∼ 0,6893484411701901
P+3Q	9	21658	196438	41490	154948	11248	143700	∼ 0,6836141724725889
3P+3Q	11	267213	2875795	575096	2300699	214397	2086302	∼ 0,680664707738155
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	181	971	211	760	19	741	∼ 0,6950431034482759
3P-Q	7	1905	13631	2939	10692	474	10218	∼ 0,6893484411701901
P-3Q	9	21658	196438	41490	154948	11248	143700	∼ 0,6836141724725889
3P-3Q	11	267213	2875795	575096	2300699	214397	2086302	∼ 0,680664707738155
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	164	988	245	743	19	724	∼ 0,6927532327586207
3P-Q	7	1775	14033	3212	10821	719	10102	∼ 0,6906515957446808
P-3Q	9	21292	203236	43969	159267	15311	143956	∼ 0,6863894943574687
3P+3Q	11	264115	2987661	602030	2385631	279808	2105823	∼ 0,6839839812528051
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	168	984	202	782	14	768	∼ 0,7033943965517241
3P-Q	7	1652	14092	2666	11426	326	11100	∼ 0,7085152257240204
3P+3Q	9	17906	207566	35362	172204	7882	164322	∼ 0,7138454296857488
3P-3Q	11	206704	3114352	465660	2648692	150080	2498612	∼ 0,7209800215751577
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	164	988	245	743	19	724	∼ 0,6927532327586207
P+3Q	7	1775	14033	3212	10821	719	10102	∼ 0,6906515957446808
P-3Q	9	21292	203236	43969	159267	15311	143956	∼ 0,6863894943574687
3P-3Q	11	264115	2987661	602030	2385631	279808	2105823	∼ 0,6839839812528051
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	181	971	211	760	19	741	∼ 0,6950431034482759
P+3Q	7	1905	13631	2939	10692	474	10218	∼ 0,6893484411701901
P-3Q	9	21658	196438	41490	154948	11248	143700	∼ 0,6836141724725889
3P+3Q	11	267213	2875795	575096	2300699	214397	2086302	∼ 0,680664707738155
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	152	1000	258	742	8	734	∼ 0,6955818965517241
P+3Q	7	1550	14450	3166	11284	630	10654	∼ 0,7033239601018675
3P+3Q	9	17532	213668	40864	172804	13800	159004	∼ 0,7091658133167116
3P-3Q	11	204360	3214328	524932	2689396	243996	2445400	∼ 0,7174826937752099
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P+Q	5	158	994	242	752	8	744	∼ 0,6974676724137931
P-3Q	7	1588	14316	2976	11340	532	10808	∼ 0,705110497237569
3P+3Q	9	17360	211696	38474	173222	11664	161558	∼ 0,7118326786591083
3P-3Q	11	197872	3189264	495152	2694112	208342	2485770	∼ 0,7209099975204925

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P-Q	5	181	971	211	760	19	741	∼ 0,6950431034482759
P+3Q	7	1905	13631	2939	10692	474	10218	∼ 0,6893484411701901
P-3Q	9	21658	196438	41490	154948	11248	143700	∼ 0,6836141724725889
3P-3Q	11	267213	2875795	575096	2300699	214397	2086302	∼ 0,680664707738155
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P-Q	5	164	988	245	743	19	724	∼ 0,6927532327586207
P+3Q	7	1775	14033	3212	10821	719	10102	∼ 0,6906515957446808
P-3Q	9	21292	203236	43969	159267	15311	143956	∼ 0,6863894943574687
3P+3Q	11	264115	2987661	602030	2385631	279808	2105823	∼ 0,6839839812528051
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P-Q	5	158	994	242	752	8	744	∼ 0,6974676724137931
P+3Q	7	1588	14316	2976	11340	532	10808	∼ 0,705110497237569
3P+3Q	9	17360	211696	38474	173222	11664	161558	∼ 0,7118326786591083
3P-3Q	11	197872	3189264	495152	2694112	208342	2485770	∼ 0,7209099975204925
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
3P-Q	5	152	1000	258	742	8	734	∼ 0,6955818965517241
P-3Q	7	1550	14450	3166	11284	630	10654	∼ 0,7033239601018675
3P+3Q	9	17532	213668	40864	172804	13800	159004	∼ 0,7091658133167116
3P-3Q	11	204360	3214328	524932	2689396	243996	2445400	∼ 0,7174826937752099
P+Q	1	2	6	4	2	0	2	∼ 0,75
P-Q	3	24	72	16	56	4	52	∼ 0,7000000000000001
P+3Q	5	168	984	202	782	14	768	∼ 0,7033943965517241
P-3Q	7	1652	14092	2666	11426	326	11100	∼ 0,7085152257240204
3P+3Q	9	17906	207566	35362	172204	7882	164322	∼ 0,7138454296857488
3P-3Q	11	206704	3114352	465660	2648692	150080	2498612	∼ 0,7209800215751577
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	269	883	689	194	78	116	∼ 0,5227640086206896
3P+Q	7	3098	11030	8639	2391	1274	1117	∼ 0,49504216539717083
3P-Q	9	36707	139773	107964	31809	17680	14129	∼ 0,48219433767001774
3P-3Q	11	444088	1792280	1350656	441624	241158	200466	∼ 0,4757437324673567
	13	5448054	23228426	16955619	6272807	3250732	3022075	∼ 0,4726160954305787
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	254	898	675	223	94	129	∼ 0,5304418103448275
3P+Q	7	2798	11570	8551	3019	1541	1478	∼ 0,5078701193058568
3P-Q	9	32784	152336	108733	43603	23489	20114	∼ 0,4978528213992734
P+3Q	11	397885	2039491	1393337	646154	347111	299043	∼ 0,4935654218524317
	13	4949898	27681958	18009715	9672243	5039488	4632755	∼ 0,4921749730361444
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	250	902	614	288	135	153	∼ 0,54296875
3P+Q	7	2769	11663	7689	3974	2125	1849	∼ 0,519986725184222
3P-Q	9	31721	154887	96981	57906	32278	25628	∼ 0,510064761308737
P-3Q	11	380016	2098176	1244931	853245	476072	377173	∼ 0,5054552083860091
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	246	906	535	371	81	290	∼ 0,5731411637931034
3P+Q	7	2537	11959	6009	5950	1150	4800	∼ 0,5752828605456908
3P-Q	9	27086	164258	69301	94957	15864	79093	∼ 0,5853080189173763
3P+3Q	11	298755	2329373	816425	1512948	219118	1293830	∼ 0,5980155423947041
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	54	18	6	12	∼ 0,58125
0+3Q	5	252	900	738	162	60	102	∼ 0,5188577586206896
3P+Q	7	2854	11546	9284	2262	1168	1094	∼ 0,497370988725065
P+3Q	9	33898	150838	117564	33274	18564	14710	∼ 0,48758539572054654
3P-3Q	11	415986	1997422	1500766	496656	275954	220702	∼ 0,4831529234042497
	13	5202926	26755826	19309252	7446574	3984716	3461858	∼ 0,48143713897262624

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	254	898	675	223	94	129	∼ 0,5304418103448275
3P+Q	7	2798	11570	8551	3019	1541	1478	∼ 0,5078701193058568
P+3Q	9	32784	152336	108733	43603	23489	20114	∼ 0,4978528213992734
P-3Q	11	397885	2039491	1393337	646154	347111	299043	∼ 0,4935654218524317
	13	4949898	27681958	18009715	9672243	5039488	4632755	∼ 0,4921749730361444
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	226	926	618	308	44	264	∼ 0,5638469827586207
3P+Q	7	2380	12436	7016	5420	1044	4376	∼ 0,5688874514877102
P+3Q	9	25746	173230	83476	89754	17280	72474	∼ 0,5787208289090993
3P+3Q	11	292590	2479090	1015644	1463446	271786	1191660	∼ 0,5904691722014397
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	261	891	690	201	83	118	∼ 0,5250538793103449
3P+Q	7	2959	11297	8778	2519	1399	1120	∼ 0,49777319139691506
P-3Q	9	35119	145633	111259	34374	21261	13113	∼ 0,48425525339925835
3P-3Q	11	428826	1901302	1416760	484542	307584	176958	∼ 0,4771426772367292
	13	5332907	25087925	18156206	6931719	4357920	2573799	∼ 0,47326971260540435
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	238	914	540	374	83	291	∼ 0,5747575431034483
3P+Q	7	2411	12213	6127	6086	1233	4853	∼ 0,5785032965845223
P-3Q	9	25677	169731	71516	98215	17801	80414	∼ 0,5895304943812406
3P+3Q	11	285363	2430333	854265	1576068	253607	1322461	∼ 0,6029018786365
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	256	896	578	318	54	264	∼ 0,5611530172413793
3P+Q	7	2734	11602	6500	5102	750	4352	∼ 0,5611979166666666
3P+3Q	9	29813	155819	74246	81573	10378	71195	∼ 0,5691554113441931
3P-3Q	11	331450	2161654	861753	1299901	141083	1158818	∼ 0,5801748234335832
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	261	891	690	201	83	118	∼ 0,5250538793103449
3P-Q	7	2959	11297	8778	2519	1399	1120	∼ 0,49777319139691506
P+3Q	9	35119	145633	111259	34374	21261	13113	∼ 0,48425525339925835
3P-3Q	11	428826	1901302	1416760	484542	307584	176958	∼ 0,4771426772367292
	13	5332907	25087925	18156206	6931719	4357920	2573799	∼ 0,47326971260540435
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	250	902	614	288	135	153	∼ 0,54296875
3P-Q	7	2769	11663	7689	3974	2125	1849	∼ 0,519986725184222
P+3Q	9	31721	154887	96981	57906	32278	25628	∼ 0,510064761308737
P-3Q	11	380016	2098176	1244931	853245	476072	377173	∼ 0,5054552083860091
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	238	914	540	374	83	291	∼ 0,5747575431034483
3P-Q	7	2411	12213	6127	6086	1233	4853	∼ 0,5785032965845223
P+3Q	9	25677	169731	71516	98215	17801	80414	∼ 0,5895304943812406
3P+3Q	11	285363	2430333	854265	1576068	253607	1322461	∼ 0,6029018786365
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	248	904	630	274	128	146	∼ 0,5404094827586207
3P-Q	7	2708	11756	8022	3734	2192	1542	∼ 0,5148261481802426
P-3Q	9	31440	156656	102686	53970	35040	18930	∼ 0,5019243797598467
3P-3Q	11	382168	2124328	1334794	789534	534906	254628	∼ 0,49463767354014015
	13	4787170	29202078	17594692	11607386	8000892	3606494	∼ 0,49026870321404326
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
0+3Q	5	214	938	474	464	138	326	∼ 0,5948275862068966
3P-Q	7	2108	12900	5312	7588	1848	5740	∼ 0,604731182795699
P-3Q	9	21958	184442	61394	123048	25036	98012	∼ 0,6214855383734249
3P+3Q	11	240404	2710668	727278	1983390	339240	1644150	∼ 0,6398496008201148

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	241	911	554	357	70	287	∼ 0,5715247844827587
3P-Q	7	2573	12003	6353	5650	907	4743	∼ 0,5727003569110967
3P+3Q	9	28320	163728	73697	90031	12161	77870	∼ 0,5818680863314374
3P-3Q	11	318785	2300863	866172	1434691	164008	1270683	∼ 0,5938756688261402
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	269	883	689	194	78	116	∼ 0,5227640086206896
P+3Q	7	3098	11030	8639	2391	1274	1117	∼ 0,49504216539717083
P-3Q	9	36707	139773	107964	31809	17680	14129	∼ 0,48219433767001774
3P-3Q	11	444088	1792280	1350656	441624	241158	200466	∼ 0,4757437324673567
	13	5448054	23228426	16955619	6272807	3250732	3022075	∼ 0,4726160954305787
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	246	906	535	371	81	290	∼ 0,5731411637931034
P+3Q	7	2537	11959	6009	5950	1150	4800	∼ 0,5752828605456908
P-3Q	9	27086	164258	69301	94957	15864	79093	∼ 0,5853080189173763
3P+3Q	11	298755	2329373	816425	1512948	219118	1293830	∼ 0,5980155423947041
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	256	896	578	318	54	264	∼ 0,5611530172413793
P+3Q	7	2734	11602	6500	5102	750	4352	∼ 0,5611979166666666
3P+3Q	9	29813	155819	74246	81573	10378	71195	∼ 0,5691554113441931
3P-3Q	11	331450	2161654	861753	1299901	141083	1158818	∼ 0,5801748234335832
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	241	911	554	357	70	287	∼ 0,5715247844827587
P-3Q	7	2573	12003	6353	5650	907	4743	∼ 0,5727003569110967
3P+3Q	9	28320	163728	73697	90031	12161	77870	∼ 0,5818680863314374
3P-3Q	11	318785	2300863	866172	1434691	164008	1270683	∼ 0,5938756688261402
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	232	920	683	237	109	128	∼ 0,5351562500000001
3P-Q	7	2670	12050	8781	3269	1823	1446	∼ 0,512325939119171
P+3Q	9	32085	160715	113644	47071	27464	19607	∼ 0,501618680508809
3P-3Q	11	400274	2171166	1480408	690758	400294	290464	∼ 0,496538133718615
	13	5098802	29639854	19417096	10222758	5735437	4487321	∼ 0,49439175256318285
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	246	906	610	296	141	155	∼ 0,544854525862069
3P-Q	7	2665	11831	7633	4198	2217	1981	∼ 0,5244085643135556
P+3Q	9	30933	158363	97355	61008	32862	28146	∼ 0,5151895291592296
P-3Q	11	373049	2160759	1251354	909405	485261	424144	∼ 0,511692928208792
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	217	935	545	390	93	297	∼ 0,5805495689655172
3P-Q	7	2182	12778	6231	6547	1528	5019	∼ 0,5872807725414245
P+3Q	9	23396	181052	73737	107315	23257	84058	∼ 0,5998497216802479
3P+3Q	11	264274	2632558	897793	1734765	341718	1393047	∼ 0,6139826495493711
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	239	913	646	267	118	149	∼ 0,541082974137931
3P-Q	7	2681	11927	8093	3834	2196	1638	∼ 0,5179256756756757
P-3Q	9	31275	159557	103882	55675	33714	21961	∼ 0,5068993884343687
3P-3Q	11	380798	2172114	1343505	828609	507897	320712	∼ 0,5018180485848113
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	205	947	509	438	113	325	∼ 0,5924030172413793
3P-Q	7	2022	13130	5733	7397	1805	5592	∼ 0,6028050547327752
P-3Q	9	21483	188597	67201	121396	26234	95162	∼ 0,6188119482400711
3P+3Q	11	239513	2778039	808304	1969735	375690	1594045	∼ 0,6360777604483151

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	224	928	551	377	83	294	∼ 0,5774515086206896
3P-Q	7	2364	12484	6353	6131	1221	4910	∼ 0,5811624461206897
3P+3Q	9	26125	173619	75019	98600	17163	81437	∼ 0,5916650811770675
3P-3Q	11	298048	2479856	901085	1578771	240305	1338466	∼ 0,6044504422230805
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	246	906	731	175	56	119	∼ 0,5237068965517241
P+3Q	7	2960	11536	9156	2380	1220	1160	∼ 0,49899848419229104
P-3Q	9	35272	149304	116669	32635	18518	14117	∼ 0,48615216449709686
3P-3Q	11	433259	1955605	1483333	472272	274261	198011	∼ 0,479915764289408
	13	5394407	25895273	18987221	6908052	3921852	2986200	∼ 0,4769025276205372
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	239	913	575	338	54	284	∼ 0,5688308189655172
P+3Q	7	2492	12116	6427	5689	1053	4636	∼ 0,57225
P-3Q	9	26736	167120	75717	91403	15418	75985	∼ 0,5814069104463746
3P+3Q	11	299627	2374293	908905	1465388	227667	1237721	∼ 0,5929678883361295
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
3P+Q	5	228	924	608	316	50	266	∼ 0,5649245689655172
P+3Q	7	2474	12310	7001	5309	924	4385	∼ 0,5674409512510785
3P+3Q	9	27427	169533	83166	86367	14626	71741	∼ 0,5755104149804593
3P-3Q	11	314699	2397829	1008967	1388862	217962	1170900	∼ 0,5860257641911558
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	229	923	571	352	66	286	∼ 0,572332974137931
P-3Q	7	2484	12284	6577	5707	1016	4691	∼ 0,5740898058252427
3P+3Q	9	27355	169189	77750	91439	14637	76802	∼ 0,5827450430336403
3P-3Q	11	310802	2396222	935348	1460874	209140	1251734	∼ 0,5940086697550012
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P-Q	5	256	896	650	246	106	140	∼ 0,5347521551724137
P+3Q	7	2943	11393	8115	3278	1866	1412	∼ 0,5075344509548612
P-3Q	9	34153	148135	103325	44810	28259	16551	∼ 0,49282418557112817
3P-3Q	11	413027	1957133	1321879	635254	419801	215453	∼ 0,4846839776076754
	13	5113997	26200131	17127470	9072661	6060634	3012027	∼ 0,4799279182226816
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	213	939	544	395	89	306	∼ 0,5829741379310345
P+3Q	7	2242	12782	6213	6569	1494	5075	∼ 0,5881329284024941
P-3Q	9	24181	180331	73724	106607	22265	84342	∼ 0,5995650702997775
3P+3Q	11	273330	2611966	894772	1717194	328572	1388622	∼ 0,6128480563935617
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	230	922	576	346	64	282	∼ 0,5708512931034483
P+3Q	7	2504	12248	6647	5601	976	4625	∼ 0,5722580384117393
3P+3Q	9	27571	168397	78213	90184	14481	75703	∼ 0,5808617985461617
3P-3Q	11	313453	2380899	937663	1443236	210489	1232747	∼ 0,5919123254221242
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	213	939	544	395	85	310	∼ 0,5835129310344828
P-3Q	7	2240	12784	6238	6546	1432	5114	∼ 0,5885159105568695
3P+3Q	9	24437	180107	74379	105728	21153	84575	∼ 0,599418004587156
3P-3Q	11	278761	2602951	905239	1697712	309508	1388204	∼ 0,6122633358132732
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	261	891	583	308	36	272	∼ 0,5602101293103449
P-3Q	7	2932	11324	6355	4969	608	4361	∼ 0,5585962958939822
3P+3Q	9	31204	149980	71359	78621	8074	70547	∼ 0,5654987819517018
3P-3Q	11	339854	2059826	811906	1247920	110111	1137809	∼ 0,5758653592636658

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	246	906	731	175	56	119	∼ 0,5237068965517241
3P-Q	7	2960	11536	9156	2380	1220	1160	∼ 0,49899848419229104
P+3Q	9	35272	149304	116669	32635	18518	14117	∼ 0,48615216449709686
3P-3Q	11	433259	1955605	1483333	472272	274261	198011	∼ 0,479915764289408
	13	5394407	25895273	18987221	6908052	3921852	2986200	∼ 0,4769025276205372
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	246	906	610	296	141	155	∼ 0,544854525862069
3P-Q	7	2665	11831	7633	4198	2217	1981	∼ 0,5244085643135556
P+3Q	9	30933	158363	97355	61008	32862	28146	∼ 0,5151895291592296
P-3Q	11	373049	2160759	1251354	909405	485261	424144	∼ 0,511692928208792
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	239	913	575	338	54	284	∼ 0,5688308189655172
3P-Q	7	2492	12116	6427	5689	1053	4636	∼ 0,57225
P+3Q	9	26736	167120	75717	91403	15418	75985	∼ 0,5814069104463746
3P+3Q	11	299627	2374293	908905	1465388	227667	1237721	∼ 0,5929678883361295
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	256	896	650	246	106	140	∼ 0,5347521551724137
3P-Q	7	2943	11393	8115	3278	1866	1412	∼ 0,5075344509548612
P-3Q	9	34153	148135	103325	44810	28259	16551	∼ 0,49282418557112817
3P-3Q	11	413027	1957133	1321879	635254	419801	215453	∼ 0,4846839776076754
	13	5113997	26200131	17127470	9072661	6060634	3012027	∼ 0,4799279182226816
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	213	939	544	395	89	306	∼ 0,5829741379310345
3P-Q	7	2242	12782	6213	6569	1494	5075	∼ 0,5881329284024941
P-3Q	9	24181	180331	73724	106607	22265	84342	∼ 0,5995650702997775
3P+3Q	11	273330	2611966	894772	1717194	328572	1388622	∼ 0,6128480563935617
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	261	891	583	308	36	272	∼ 0,5602101293103449
3P-Q	7	2932	11324	6355	4969	608	4361	∼ 0,5585962958939822
3P+3Q	9	31204	149980	71359	78621	8074	70547	∼ 0,5654987819517018
3P-3Q	11	339854	2059826	811906	1247920	110111	1137809	∼ 0,5758653592636658
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	232	920	683	237	109	128	∼ 0,5351562500000001
P+3Q	7	2670	12050	8781	3269	1823	1446	∼ 0,512325939119171
P-3Q	9	32085	160715	113644	47071	27464	19607	∼ 0,501618680508809
3P-3Q	11	400274	2171166	1480408	690758	400294	290464	∼ 0,496538133718615
	13	5098802	29639854	19417096	10222758	5735437	4487321	∼ 0,49439175256318285
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	217	935	545	390	93	297	∼ 0,5805495689655172
P+3Q	7	2182	12778	6231	6547	1528	5019	∼ 0,5872807725414245
P-3Q	9	23396	181052	73737	107315	23257	84058	∼ 0,5998497216802479
3P+3Q	11	264274	2632558	897793	1734765	341718	1393047	∼ 0,6139826495493711
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	6	16	∼ 0,59375
3P+Q	5	228	924	608	316	50	266	∼ 0,5649245689655172
P+3Q	7	2474	12310	7001	5309	924	4385	∼ 0,5674409512510785
3P+3Q	9	27427	169533	83166	86367	14626	71741	∼ 0,5755104149804593
3P-3Q	11	314699	2397829	1008967	1388862	217962	1170900	∼ 0,5860257641911558
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	230	922	576	346	64	282	∼ 0,5708512931034483
P-3Q	7	2504	12248	6647	5601	976	4625	∼ 0,5722580384117393
3P+3Q	9	27571	168397	78213	90184	14481	75703	∼ 0,5808617985461617
3P-3Q	11	313453	2380899	937663	1443236	210489	1232747	∼ 0,5919123254221242

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P-Q	5	239	913	646	267	118	149	∼ 0,541082974137931
P+3Q	7	2681	11927	8093	3834	2196	1638	∼ 0,5179256756756757
P-3Q	9	31275	159557	103882	55675	33714	21961	∼ 0,5068993884343687
3P-3Q	11	380798	2172114	1343505	828609	507897	320712	∼ 0,5018180485848113
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	205	947	509	438	113	325	∼ 0,5924030172413793
P+3Q	7	2022	13130	5733	7397	1805	5592	∼ 0,6028050547327752
P-3Q	9	21483	188597	67201	121396	26234	95162	∼ 0,6188119482400711
3P+3Q	11	239513	2778039	808304	1969735	375690	1594045	∼ 0,6360777604483151
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	229	923	571	352	66	286	∼ 0,572332974137931
P+3Q	7	2484	12284	6577	5707	1016	4691	∼ 0,5740898058252427
3P+3Q	9	27355	169189	77750	91439	14637	76802	∼ 0,5827450430336403
3P-3Q	11	310802	2396222	935348	1460874	209140	1251734	∼ 0,5940086697550012
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	213	939	544	395	85	310	∼ 0,5835129310344828
P-3Q	7	2240	12784	6238	6546	1432	5114	∼ 0,5885159105568695
3P+3Q	9	24437	180107	74379	105728	21153	84575	∼ 0,599418004587156
3P-3Q	11	278761	2602951	905239	1697712	309508	1388204	∼ 0,6122633358132732
P+Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	224	928	551	377	83	294	∼ 0,5774515086206896
P-3Q	7	2364	12484	6353	6131	1221	4910	∼ 0,5811624461206897
3P+3Q	9	26125	173619	75019	98600	17163	81437	∼ 0,5916650811770675
3P-3Q	11	298048	2479856	901085	1578771	240305	1338466	∼ 0,6044504422230805
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	50	22	8	14	∼ 0,59375
3P-Q	5	224	928	658	270	124	146	∼ 0,5431034482758621
P+3Q	7	2892	11956	8674	3282	1748	1534	∼ 0,5129445043103448
P-3Q	9	36862	154434	114708	39726	24676	15050	∼ 0,49172162861385676
3P-3Q	11	479868	1991076	1490244	500832	350308	150524	∼ 0,47757886826163787
	13	6122218	25734998	19240090	6494908	4882326	1612582	∼ 0,4684109126757055
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	198	954	558	396	94	302	∼ 0,5845905172413793
P+3Q	7	2328	12936	6652	6284	1748	4536	∼ 0,5797524646378054
P-3Q	9	27742	179234	82600	96634	26196	70438	∼ 0,5802862989770755
3P+3Q	11	333956	2533788	1038452	1495336	386686	1108650	∼ 0,5844049272919822
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	209	943	621	322	44	278	∼ 0,5699084051724138
P+3Q	7	2662	12426	7304	5122	861	4261	∼ 0,5650241675617615
3P+3Q	9	31406	167410	87508	79902	13132	66770	∼ 0,5666949059903804
3P-3Q	11	366656	2311904	1062193	1249711	189564	1060147	∼ 0,5723707616184802
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	202	950	593	357	67	290	∼ 0,5771821120689655
P-3Q	7	2468	12732	7036	5696	1343	4353	∼ 0,5720586658086658
3P+3Q	9	29199	174513	86977	87536	19962	67574	∼ 0,5725126397095873
3P-3Q	11	352324	2439884	1076333	1363551	294235	1069316	∼ 0,5772730645601538
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	209	943	621	322	44	278	∼ 0,5699084051724138
P-3Q	7	2662	12426	7304	5122	861	4261	∼ 0,5650241675617615
3P+3Q	9	31406	167410	87508	79902	13132	66770	∼ 0,5666949059903804
3P-3Q	11	366656	2311904	1062193	1249711	189564	1060147	∼ 0,5723707616184802
P+Q	1	2	6	5	1	0	1	∼ 0,75
3P-Q	3	24	72	46	26	8	18	∼ 0,60625
P+3Q	5	202	950	593	357	67	290	∼ 0,5771821120689655
P-3Q	7	2468	12732	7036	5696	1343	4353	∼ 0,5720586658086658
3P+3Q	9	29199	174513	86977	87536	19962	67574	∼ 0,5725126397095873
3P-3Q	11	352324	2439884	1076333	1363551	294235	1069316	∼ 0,5772730645601538

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	246	906	535	371	81	290	∼ 0,5731411637931034
3P+Q	7	2537	11959	6009	5950	1150	4800	∼ 0,5752828605456908
3P-Q	9	27086	164258	69301	94957	15864	79093	∼ 0,5853080189173763
3P-3Q	11	298755	2329373	816425	1512948	219118	1293830	∼ 0,5980155423947041
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	250	902	614	288	135	153	∼ 0,54296875
3P+Q	7	2769	11663	7689	3974	2125	1849	∼ 0,519986725184222
3P-Q	9	31721	154887	96981	57906	32278	25628	∼ 0,510064761308737
P+3Q	11	380016	2098176	1244931	853245	476072	377173	∼ 0,5054552083860091
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	254	898	675	223	94	129	∼ 0,5304418103448275
3P+Q	7	2798	11570	8551	3019	1541	1478	∼ 0,5078701193058568
3P-Q	9	32784	152336	108733	43603	23489	20114	∼ 0,4978528213992734
P-3Q	11	397885	2039491	1393337	646154	347111	299043	∼ 0,4935654218524317
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	269	883	689	194	78	116	∼ 0,5227640086206896
3P+Q	7	3098	11030	8639	2391	1274	1117	∼ 0,49504216539717083
3P-Q	9	36707	139773	107964	31809	17680	14129	∼ 0,48219433767001774
3P+3Q	11	444088	1792280	1350656	441624	241158	200466	∼ 0,4757437324673567
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
0+3Q	5	214	938	474	464	138	326	∼ 0,5948275862068966
3P+Q	7	2108	12900	5312	7588	1848	5740	∼ 0,604731182795699
P+3Q	9	21958	184442	61394	123048	25036	98012	∼ 0,6214855383734249
3P-3Q	11	240404	2710668	727278	1983390	339240	1644150	∼ 0,6398496008201148
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	250	902	614	288	135	153	∼ 0,54296875
3P+Q	7	2769	11663	7689	3974	2125	1849	∼ 0,519986725184222
P+3Q	9	31721	154887	96981	57906	32278	25628	∼ 0,510064761308737
P-3Q	11	380016	2098176	1244931	853245	476072	377173	∼ 0,5054552083860091
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
0+3Q	5	248	904	630	274	128	146	∼ 0,5404094827586207
3P+Q	7	2708	11756	8022	3734	2192	1542	∼ 0,5148261481802426
P+3Q	9	31440	156656	102686	53970	35040	18930	∼ 0,5019243797598467
3P+3Q	11	382168	2124328	1334794	789534	534906	254628	∼ 0,49463767354014015
	13	4787170	29202078	17594692	11607386	8000892	3606494	∼ 0,49026870321404326
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	238	914	540	374	83	291	∼ 0,5747575431034483
3P+Q	7	2411	12213	6127	6086	1233	4853	∼ 0,5785032965845223
P-3Q	9	25677	169731	71516	98215	17801	80414	∼ 0,5895304943812406
3P-3Q	11	285363	2430333	854265	1576068	253607	1322461	∼ 0,6029018786365
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	261	891	690	201	83	118	∼ 0,5250538793103449
3P+Q	7	2959	11297	8778	2519	1399	1120	∼ 0,49777319139691506
P-3Q	9	35119	145633	111259	34374	21261	13113	∼ 0,48425525339925835
3P+3Q	11	428826	1901302	1416760	484542	307584	176958	∼ 0,4771426772367292
	13	5332907	25087925	18156206	6931719	4357920	2573799	∼ 0,47326971260540435
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	241	911	554	357	70	287	∼ 0,5715247844827587
3P+Q	7	2573	12003	6353	5650	907	4743	∼ 0,5727003569110967
3P+3Q	9	28320	163728	73697	90031	12161	77870	∼ 0,5818680863314374
3P-3Q	11	318785	2300863	866172	1434691	164008	1270683	∼ 0,5938756688261402

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	238	914	540	374	83	291	∼ 0,5747575431034483
3P-Q	7	2411	12213	6127	6086	1233	4853	∼ 0,5785032965845223
P+3Q	9	25677	169731	71516	98215	17801	80414	∼ 0,5895304943812406
3P-3Q	11	285363	2430333	854265	1576068	253607	1322461	∼ 0,6029018786365
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	254	898	675	223	94	129	∼ 0,5304418103448275
3P-Q	7	2798	11570	8551	3019	1541	1478	∼ 0,5078701193058568
P+3Q	9	32784	152336	108733	43603	23489	20114	∼ 0,4978528213992734
P-3Q	11	397885	2039491	1393337	646154	347111	299043	∼ 0,4935654218524317
	13	4949898	27681958	18009715	9672243	5039488	4632755	∼ 0,4921749730361444
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	261	891	690	201	83	118	∼ 0,5250538793103449
3P-Q	7	2959	11297	8778	2519	1399	1120	∼ 0,49777319139691506
P+3Q	9	35119	145633	111259	34374	21261	13113	∼ 0,48425525339925835
3P+3Q	11	428826	1901302	1416760	484542	307584	176958	∼ 0,4771426772367292
	13	5332907	25087925	18156206	6931719	4357920	2573799	∼ 0,47326971260540435
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	226	926	618	308	44	264	∼ 0,5638469827586207
3P-Q	7	2380	12436	7016	5420	1044	4376	∼ 0,5688874514877102
P-3Q	9	25746	173230	83476	89754	17280	72474	∼ 0,5787208289090993
3P-3Q	11	292590	2479090	1015644	1463446	271786	1191660	∼ 0,5904691722014397
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	54	18	6	12	∼ 0,58125
0+3Q	5	252	900	738	162	60	102	∼ 0,5188577586206896
3P-Q	7	2854	11546	9284	2262	1168	1094	∼ 0,497370988725065
P-3Q	9	33898	150838	117564	33274	18564	14710	∼ 0,48758539572054654
3P+3Q	11	415986	1997422	1500766	496656	275954	220702	∼ 0,4831529234042497
	13	5202926	26755826	19309252	7446574	3984716	3461858	∼ 0,48143713897262624
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	256	896	578	318	54	264	∼ 0,5611530172413793
3P-Q	7	2734	11602	6500	5102	750	4352	∼ 0,5611979166666666
3P+3Q	9	29813	155819	74246	81573	10378	71195	∼ 0,5691554113441931
3P-3Q	11	331450	2161654	861753	1299901	141083	1158818	∼ 0,5801748234335832
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	246	906	535	371	81	290	∼ 0,5731411637931034
P+3Q	7	2537	11959	6009	5950	1150	4800	∼ 0,5752828605456908
P-3Q	9	27086	164258	69301	94957	15864	79093	∼ 0,5853080189173763
3P-3Q	11	298755	2329373	816425	1512948	219118	1293830	∼ 0,5980155423947041
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
0+3Q	5	269	883	689	194	78	116	∼ 0,5227640086206896
P+3Q	7	3098	11030	8639	2391	1274	1117	∼ 0,49504216539717083
P-3Q	9	36707	139773	107964	31809	17680	14129	∼ 0,48219433767001774
3P+3Q	11	444088	1792280	1350656	441624	241158	200466	∼ 0,4757437324673567
	13	5448054	23228426	16955619	6272807	3250732	3022075	∼ 0,4726160954305787
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
0+3Q	5	241	911	554	357	70	287	∼ 0,5715247844827587
P+3Q	7	2573	12003	6353	5650	907	4743	∼ 0,5727003569110967
3P+3Q	9	28320	163728	73697	90031	12161	77870	∼ 0,5818680863314374
3P-3Q	11	318785	2300863	866172	1434691	164008	1270683	∼ 0,5938756688261402
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
0+3Q	5	256	896	578	318	54	264	∼ 0,5611530172413793
P-3Q	7	2734	11602	6500	5102	750	4352	∼ 0,5611979166666666
3P+3Q	9	29813	155819	74246	81573	10378	71195	∼ 0,5691554113441931
3P-3Q	11	331450	2161654	861753	1299901	141083	1158818	∼ 0,5801748234335832

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	205	947	509	438	113	325	∼ 0,5924030172413793
3P-Q	7	2022	13130	5733	7397	1805	5592	∼ 0,6028050547327752
P+3Q	9	21483	188597	67201	121396	26234	95162	∼ 0,6188119482400711
3P-3Q	11	239513	2778039	808304	1969735	375690	1594045	∼ 0,6360777604483151
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	246	906	610	296	141	155	∼ 0,544854525862069
3P-Q	7	2665	11831	7633	4198	2217	1981	∼ 0,5244085643135556
P+3Q	9	30933	158363	97355	61008	32862	28146	∼ 0,5151895291592296
P-3Q	11	373049	2160759	1251354	909405	485261	424144	∼ 0,511692928208792
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	239	913	646	267	118	149	∼ 0,541082974137931
3P-Q	7	2681	11927	8093	3834	2196	1638	∼ 0,5179256756756757
P+3Q	9	31275	159557	103882	55675	33714	21961	∼ 0,5068993884343687
3P+3Q	11	380798	2172114	1343505	828609	507897	320712	∼ 0,5018180485848113
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	217	935	545	390	93	297	∼ 0,5805495689655172
3P-Q	7	2182	12778	6231	6547	1528	5019	∼ 0,5872807725414245
P-3Q	9	23396	181052	73737	107315	23257	84058	∼ 0,5998497216802479
3P-3Q	11	264274	2632558	897793	1734765	341718	1393047	∼ 0,6139826495493711
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	232	920	683	237	109	128	∼ 0,5351562500000001
3P-Q	7	2670	12050	8781	3269	1823	1446	∼ 0,512325939119171
P-3Q	9	32085	160715	113644	47071	27464	19607	∼ 0,501618680508809
3P+3Q	11	400274	2171166	1480408	690758	400294	290464	∼ 0,496538133718615
	13	5098802	29639854	19417096	10222758	5735437	4487321	∼ 0,49439175256318285
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	224	928	551	377	83	294	∼ 0,5774515086206896
3P-Q	7	2364	12484	6353	6131	1221	4910	∼ 0,5811624461206897
3P+3Q	9	26125	173619	75019	98600	17163	81437	∼ 0,5916650811770675
3P-3Q	11	298048	2479856	901085	1578771	240305	1338466	∼ 0,6044504422230805
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	213	939	544	395	89	306	∼ 0,5829741379310345
P+3Q	7	2242	12782	6213	6569	1494	5075	∼ 0,5881329284024941
P-3Q	9	24181	180331	73724	106607	22265	84342	∼ 0,5995650702997775
3P-3Q	11	273330	2611966	894772	1717194	328572	1388622	∼ 0,6128480563935617
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	256	896	650	246	106	140	∼ 0,5347521551724137
P+3Q	7	2943	11393	8115	3278	1866	1412	∼ 0,5075344509548612
P-3Q	9	34153	148135	103325	44810	28259	16551	∼ 0,49282418557112817
3P+3Q	11	413027	1957133	1321879	635254	419801	215453	∼ 0,4846839776076754
	13	5113997	26200131	17127470	9072661	6060634	3012027	∼ 0,4799279182226816
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	213	939	544	395	85	310	∼ 0,5835129310344828
P+3Q	7	2240	12784	6238	6546	1432	5114	∼ 0,5885159105568695
3P+3Q	9	24437	180107	74379	105728	21153	84575	∼ 0,599418004587156
3P-3Q	11	278761	2602951	905239	1697712	309508	1388204	∼ 0,6122633358132732
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	230	922	576	346	64	282	∼ 0,5708512931034483
P-3Q	7	2504	12248	6647	5601	976	4625	∼ 0,5722580384117393
3P+3Q	9	27571	168397	78213	90184	14481	75703	∼ 0,5808617985461617
3P-3Q	11	313453	2380899	937663	1443236	210489	1232747	∼ 0,5919123254221242

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	239	913	575	338	54	284	∼ 0,5688308189655172
P+3Q	7	2492	12116	6427	5689	1053	4636	∼ 0,57225
P-3Q	9	26736	167120	75717	91403	15418	75985	∼ 0,5814069104463746
3P-3Q	11	299627	2374293	908905	1465388	227667	1237721	∼ 0,5929678883361295
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	52	20	7	13	∼ 0,5875
3P-Q	5	246	906	731	175	56	119	∼ 0,5237068965517241
P+3Q	7	2960	11536	9156	2380	1220	1160	∼ 0,49899848419229104
P-3Q	9	35272	149304	116669	32635	18518	14117	∼ 0,48615216449709686
3P+3Q	11	433259	1955605	1483333	472272	274261	198011	∼ 0,479915764289408
	13	5394407	25895273	18987221	6908052	3921852	2986200	∼ 0,4769025276205372
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	229	923	571	352	66	286	∼ 0,572332974137931
P+3Q	7	2484	12284	6577	5707	1016	4691	∼ 0,5740898058252427
3P+3Q	9	27355	169189	77750	91439	14637	76802	∼ 0,5827450430336403
3P-3Q	11	310802	2396222	935348	1460874	209140	1251734	∼ 0,5940086697550012
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	50	22	6	16	∼ 0,59375
3P-Q	5	228	924	608	316	50	266	∼ 0,5649245689655172
P-3Q	7	2474	12310	7001	5309	924	4385	∼ 0,5674409512510785
3P+3Q	9	27427	169533	83166	86367	14626	71741	∼ 0,5755104149804593
3P-3Q	11	314699	2397829	1008967	1388862	217962	1170900	∼ 0,5860257641911558
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+0	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	261	891	583	308	36	272	∼ 0,5602101293103449
P-3Q	7	2932	11324	6355	4969	608	4361	∼ 0,5585962958939822
3P+3Q	9	31204	149980	71359	78621	8074	70547	∼ 0,5654987819517018
3P-3Q	11	339854	2059826	811906	1247920	110111	1137809	∼ 0,5758653592636658
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	213	939	544	395	89	306	∼ 0,5829741379310345
3P-Q	7	2242	12782	6213	6569	1494	5075	∼ 0,5881329284024941
P+3Q	9	24181	180331	73724	106607	22265	84342	∼ 0,5995650702997775
3P-3Q	11	273330	2611966	894772	1717194	328572	1388622	∼ 0,6128480563935617
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	246	906	610	296	141	155	∼ 0,5448545258620 69
3P-Q	7	2665	11831	7633	4198	2217	1981	∼ 0,5244085643135556
P+3Q	9	30933	158363	97355	61008	32862	28146	∼ 0,5151895291592296
P-3Q	11	373049	2160759	1251354	909405	485261	424144	∼ 0,511692928208792
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	256	896	650	246	106	140	∼ 0,5347521551724137
3P-Q	7	2943	11393	8115	3278	1866	1412	∼ 0,5075344509548612
P+3Q	9	34153	148135	103325	44810	28259	16551	∼ 0,49282418557112817
3P+3Q	11	413027	1957133	1321879	635254	419801	215453	∼ 0,4846839776076754
	13	5113997	26200131	17127470	9072661	6060634	3012027	∼ 0,4799279182226816
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	239	913	575	338	54	284	∼ 0,5688308189655172
3P-Q	7	2492	12116	6427	5689	1053	4636	∼ 0,57225
P-3Q	9	26736	167120	75717	91403	15418	75985	∼ 0,5814069104463746
3P-3Q	11	299627	2374293	908905	1465388	227667	1237721	∼ 0,5929678883361295
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	52	20	7	13	∼ 0,5875
3P+Q	5	246	906	731	175	56	119	∼ 0,5237068965517241
3P-Q	7	2960	11536	9156	2380	1220	1160	∼ 0,49899848419229104
P-3Q	9	35272	149304	116669	32635	18518	14117	∼ 0,48615216449709686
3P+3Q	11	433259	1955605	1483333	472272	274261	198011	∼ 0,479915764289408
	13	5394407	25895273	18987221	6908052	3921852	2986200	∼ 0,4769025276205372

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	261	891	583	308	36	272	∼ 0,5602101293103449
3P-Q	7	2932	11324	6355	4969	608	4361	∼ 0,5585962958939822
3P+3Q	9	31204	149980	71359	78621	8074	70547	∼ 0,5654987819517018
3P-3Q	11	339854	2059826	811906	1247920	110111	1137809	∼ 0,5758653592636658
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	205	947	509	438	113	325	∼ 0,5924030172413793
P+3Q	7	2022	13130	5733	7397	1805	5592	∼ 0,6028050547327752
P-3Q	9	21483	188597	67201	121396	26234	95162	∼ 0,6188119482400711
3P-3Q	11	239513	2778039	808304	1969735	375690	1594045	∼ 0,6360777604483151
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	8	14	∼ 0,59375
3P+Q	5	239	913	646	267	118	149	∼ 0,541082974137931
P+3Q	7	2681	11927	8093	3834	2196	1638	∼ 0,5179256756756757
P-3Q	9	31275	159557	103882	55675	33714	21961	∼ 0,5068993884343687
3P+3Q	11	380798	2172114	1343505	828609	507897	320712	∼ 0,5018180485848113
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	46	26	8	18	∼ 0,60625
3P+Q	5	213	939	544	395	85	310	∼ 0,5835129310344828
P+3Q	7	2240	12784	6238	6546	1432	5114	∼ 0,5885159105568695
3P+3Q	9	24437	180107	74379	105728	21153	84575	∼ 0,599418004587156
3P-3Q	11	278761	2602951	905239	1697712	309508	1388204	∼ 0,6122633358132732
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P+Q	5	229	923	571	352	66	286	∼ 0,572332974137931
P-3Q	7	2484	12284	6577	5707	1016	4691	∼ 0,5740898058252427
3P+3Q	9	27355	169189	77750	91439	14637	76802	∼ 0,5827450430336403
3P-3Q	11	310802	2396222	935348	1460874	209140	1251734	∼ 0,5940086697550012
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	217	935	545	390	93	297	∼ 0,5805495689655172
P+3Q	7	2182	12778	6231	6547	1528	5019	∼ 0,5872807725414245
P-3Q	9	23396	181052	73737	107315	23257	84058	∼ 0,5998497216802479
3P-3Q	11	264274	2632558	897793	1734765	341718	1393047	∼ 0,6139826495493711
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	52	20	7	13	∼ 0,5875
3P-Q	5	232	920	683	237	109	128	∼ 0,5351562500000001
P+3Q	7	2670	12050	8781	3269	1823	1446	∼ 0,512325939119171
P-3Q	9	32085	160715	113644	47071	27464	19607	∼ 0,501618680508809
3P+3Q	11	400274	2171166	1480408	690758	400294	290464	∼ 0,496538133718615
	13	5098802	29639854	19417096	10222758	5735437	4487321	∼ 0,49439175256318285
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	230	922	576	346	64	282	∼ 0,5708512931034483
P+3Q	7	2504	12248	6647	5601	976	4625	∼ 0,5722580384117393
3P+3Q	9	27571	168397	78213	90184	14481	75703	∼ 0,5808617985461617
3P-3Q	11	313453	2380899	937663	1443236	210489	1232747	∼ 0,5919123254221242
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	50	22	6	16	∼ 0,59375
3P-Q	5	228	924	608	316	50	266	∼ 0,5649245689655172
P-3Q	7	2474	12310	7001	5309	924	4385	∼ 0,5674409512510785
3P+3Q	9	27427	169533	83166	86367	14626	71741	∼ 0,5755104149804593
3P-3Q	11	314699	2397829	1008967	1388862	217962	1170900	∼ 0,5860257641911558
P-Q	1	2	6	5	1	0	1	∼ 0,75
0+3Q	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	224	928	551	377	83	294	∼ 0,5774515086206896
P-3Q	7	2364	12484	6353	6131	1221	4910	∼ 0,5811624461206897
3P+3Q	9	26125	173619	75019	98600	17163	81437	∼ 0,5916650811770675
3P-3Q	11	298048	2479856	901085	1578771	240305	1338466	∼ 0,6044504422230805
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	198	954	558	396	94	302	∼ 0,5845905172413793
P+3Q	7	2328	12936	6652	6284	1748	4536	∼ 0,5797524646378054
P-3Q	9	27742	179234	82600	96634	26196	70438	∼ 0,5802862989770755
3P-3Q	11	333956	2533788	1038452	1495336	386686	1108650	∼ 0,5844049272919822

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	50	22	8	14	∼ 0,59375
3P-Q	5	224	928	658	270	124	146	∼ 0,5431034482758621
P+3Q	7	2892	11956	8674	3282	1748	1534	∼ 0,5129445043103448
P-3Q	9	36862	154434	114708	39726	24676	15050	∼ 0,49172162861385676
3P+3Q	11	479868	1991076	1490244	500832	350308	150524	∼ 0,47757886826163787
	13	6122218	25734998	19240090	6494908	4882326	1612582	∼ 0,4684109126757055
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	46	26	8	18	∼ 0,60625
3P-Q	5	202	950	593	357	67	290	∼ 0,5771821120689655
P+3Q	7	2468	12732	7036	5696	1343	4353	∼ 0,5720586658086658
3P+3Q	9	29199	174513	86977	87536	19962	67574	∼ 0,5725126397095873
3P-3Q	11	352324	2439884	1076333	1363551	294235	1069316	∼ 0,5772730645601538
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	48	24	7	17	∼ 0,6
3P-Q	5	209	943	621	322	44	278	∼ 0,5699084051724138
P-3Q	7	2662	12426	7304	5122	861	4261	∼ 0,5650241675617615
3P+3Q	9	31406	167410	87508	79902	13132	66770	∼ 0,5666949059903804
3P-3Q	11	366656	2311904	1062193	1249711	189564	1060147	∼ 0,5723707616184802
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P+Q	3	24	72	46	26	8	18	∼ 0,60625
P+3Q	5	202	950	593	357	67	290	∼ 0,5771821120689655
P-3Q	7	2468	12732	7036	5696	1343	4353	∼ 0,5720586658086658
3P+3Q	9	29199	174513	86977	87536	19962	67574	∼ 0,5725126397095873
3P-3Q	11	352324	2439884	1076333	1363551	294235	1069316	∼ 0,5772730645601538
P-Q	1	2	6	5	1	0	1	∼ 0,75
3P-Q	3	24	72	48	24	7	17	∼ 0,6
P+3Q	5	209	943	621	322	44	278	∼ 0,5699084051724138
P-3Q	7	2662	12426	7304	5122	861	4261	∼ 0,5650241675617615
3P+3Q	9	31406	167410	87508	79902	13132	66770	∼ 0,5666949059903804
3P-3Q	11	366656	2311904	1062193	1249711	189564	1060147	∼ 0,5723707616184802
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	878	24	24	0	0,48679956896551724
3P-Q	7	2864	11568	10826	742	742	0	0,4724073472041612
P+3Q	9	33892	151196	136604	14592	14592	0	0,4647376217912685
3P-3Q	11	413896	2005240	1751206	254034	253914	120	∼ 0,46016081558227256
	13	5174330	26909510	22772094	4137416	4126626	10790	∼ 0,4571899840470101
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	240	912	848	64	64	0	0,49353448275862066
3P-Q	7	2688	11904	10496	1408	1408	0	0,47923875432525953
P+3Q	9	31488	158976	132864	26112	26112	0	0,4719634873323398
P-3Q	11	384000	2159616	1718272	441344	437248	4096	∼ 0,46785773918851226
	13	4826112	29727744	22604800	7122944	6967296	155648	∼ 0,4654841262229043
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	850	52	52	0	0,49057112068965525
3P-Q	7	2780	11652	10446	1206	1206	0	0,47611887732986563
P+3Q	9	32532	153900	131704	22196	22196	0	0,4685408769713136
3P+3Q	11	395328	2067072	1694166	372906	371006	1900	∼ 0,46412447355137904
	13	4944714	28128438	22153554	5974884	5891966	82918	∼ 0,461438241575736
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	850	52	52	0	0,49057112068965525
3P-Q	7	2780	11652	10446	1206	1206	0	0,47611887732986563
P-3Q	9	32532	153900	131704	22196	22196	0	0,4685408769713136
3P-3Q	11	395328	2067072	1694166	372906	371006	1900	∼ 0,46412447355137904
	13	4944714	28128438	22153554	5974884	5891966	82918	∼ 0,461438241575736
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	878	24	24	0	0,48679956896551724
3P-Q	7	2864	11568	10826	742	742	0	0,4724073472041612
P-3Q	9	33892	151196	136604	14592	14592	0	0,4647376217912685
3P+3Q	11	413896	2005240	1751206	254034	253914	120	∼ 0,46016081558227256
	13	5174330	26909510	22772094	4137416	4126626	10790	∼ 0,4571899840470101

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	284	868	856	12	12	0	0,48060344827586216
3P-Q	7	3380	10508	10292	216	216	0	0,46138100436681223
3P+3Q	9	40416	127712	124400	3312	3312	0	0,45109681577314176
3P-3Q	11	485352	1558040	1509752	48288	48288	0	0,44492055546474885
	13	5852608	19076032	18391408	684624	684624	0	0,4409356093689873
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	878	24	24	0	0,48679956896551724
P+3Q	7	2864	11568	10826	742	742	0	0,4724073472041612
P-3Q	9	33892	151196	136604	14592	14592	0	0,4647376217912685
3P-3Q	11	413896	2005240	1751206	254034	253914	120	$\sim$ 0,46016081558227256
	13	5174330	26909510	22772094	4137416	4126626	10790	$\sim$ 0,4571899840470101
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	250	902	850	52	52	0	0,49057112068965525
P+3Q	7	2780	11652	10446	1206	1206	0	0,47611887732986563
P-3Q	9	32532	153900	131704	22196	22196	0	0,4685408769713136
3P+3Q	11	395328	2067072	1694166	372906	371006	1900	$\sim$ 0,46412447355137904
	13	4944714	28128438	22153554	5974884	5891966	82918	$\sim$ 0,461438241575736
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	260	892	860	32	32	0	0,4865301724137931
P+3Q	7	2940	11332	10516	816	816	0	0,4712478279756733
3P+3Q	9	34500	146812	131372	15440	15440	0	0,46327327168337334
3P-3Q	11	417308	1931684	1670868	260816	260304	512	$\sim$ 0,4585249988717423
	13	5172964	25733980	21570892	4163088	4137744	25344	$\sim$ 0,4554944657094495
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P+Q	5	276	876	864	12	12	0	0,4816810344827586
P-3Q	7	3240	10776	10472	304	304	0	0,4638731473408893
3P+3Q	9	38556	133860	128220	5640	5640	0	0,45447323719471633
3P-3Q	11	464292	1677468	1584624	92844	92832	12	$\sim$ 0,4488465387721805
	13	5651928	21187560	19750456	1437104	1436344	760	$\sim$ 0,44521110326817953
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P-Q	5	250	902	850	52	52	0	0,49057112068965525
P+3Q	7	2780	11652	10446	1206	1206	0	0,47611887732986563
P-3Q	9	32532	153900	131704	22196	22196	0	0,4685408769713136
3P-3Q	11	395328	2067072	1694166	372906	371006	1900	$\sim$ 0,46412447355137904
	13	4944714	28128438	22153554	5974884	5891966	82918	$\sim$ 0,461438241575736
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P-Q	5	250	902	878	24	24	0	0,48679956896551724
P+3Q	7	2864	11568	10826	742	742	0	0,4724073472041612
P-3Q	9	33892	151196	136604	14592	14592	0	0,4647376217912685
3P+3Q	11	413896	2005240	1751206	254034	253914	120	$\sim$ 0,46016081558227256
	13	5174330	26909510	22772094	4137416	4126626	10790	$\sim$ 0,4571899840470101
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P-Q	5	276	876	864	12	12	0	0,4816810344827586
P+3Q	7	3240	10776	10472	304	304	0	0,4638731473408893
3P+3Q	9	38556	133860	128220	5640	5640	0	0,45447323719471633
3P-3Q	11	464292	1677468	1584624	92844	92832	12	$\sim$ 0,4488465387721805
	13	5651928	21187560	19750456	1437104	1436344	760	$\sim$ 0,44521110326817953
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
3P-Q	5	260	892	860	32	32	0	0,4865301724137931
P-3Q	7	2940	11332	10516	816	816	0	0,4712478279756733
3P+3Q	9	34500	146812	131372	15440	15440	0	0,46327327168337334
3P-3Q	11	417308	1931684	1670868	260816	260304	512	$\sim$ 0,4585249988717423
	13	5172964	25733980	21570892	4163088	4137744	25344	$\sim$ 0,4554944657094495

FP $\widehat{=}$ Fehlende Punkte bei der Rekodierung, $w \widehat{=}$ Fensterbreite in Bits, pos $\widehat{=}$ Anzahl der positiven Rekodierungen
neg $\widehat{=}$ Anzahl der negativen Rekodierungen, pos-1 $\widehat{=}$ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 $\widehat{=}$ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 $\widehat{=}$ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 $\widehat{=}$ Anzahl der negativen Rekodierungen mit Nullspalten-2, $\sim \widehat{=}$ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
3P+0	1	2	6	6	0	0	0	0,75
0+3Q	3	24	72	72	0	0	0	0,525
P+3Q	5	284	868	856	12	12	0	0,48060344827586216
P-3Q	7	3380	10508	10292	216	216	0	0,46138100436681223
3P+3Q	9	40416	127712	124400	3312	3312	0	0,45109681577314176
3P-3Q	11	485352	1558040	1509752	48288	48288	0	0,44492055546474885
	13	5852608	19076032	18391408	684624	684624	0	0,4409356093689873
3P+0	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	236	916	832	84	84	0	0,4967672413793103
P+3Q	7	2668	11988	10456	1532	1532	0	0,4806611927398444
P-3Q	9	31660	160148	132224	27924	27924	0	0,4730955646473569
3P-3Q	11	388036	2174332	1714468	459864	454880	4984	∼ 0,46867988331021054
	13	4888112	29901200	22565020	7336180	7170144	166036	∼ 0,4660883492048925
3P+0	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	236	916	832	84	84	0	0,4967672413793103
P+3Q	7	2668	11988	10456	1532	1532	0	0,4806611927398444
P-3Q	9	31660	160148	132224	27924	27924	0	0,4730955646473569
3P+3Q	11	388036	2174332	1714468	459864	454880	4984	∼ 0,46867988331021054
	13	4888112	29901200	22565020	7336180	7170144	166036	∼ 0,4660883492048925
3P+0	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	246	906	864	42	42	0	0,48976293103448276
P+3Q	7	2834	11662	10672	990	990	0	0,47474556084885233
3P+3Q	9	33610	152982	134576	18406	18406	0	0,46684833430742256
3P-3Q	11	410360	2037352	1727002	310350	309300	1050	∼ 0,4621660247657023
	13	5128440	27469192	22493872	4975320	4927890	47430	∼ 0,45922627121476856
3P+0	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	246	906	864	42	42	0	0,48976293103448276
P-3Q	7	2834	11662	10672	990	990	0	0,47474556084885233
3P+3Q	9	33610	152982	134576	18406	18406	0	0,46684833430742256
3P-3Q	11	410360	2037352	1727002	310350	309300	1050	∼ 0,4621660247657023
	13	5128440	27469192	22493872	4975320	4927890	47430	∼ 0,45922627121476856
3P+0	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
P+3Q	5	262	890	852	38	38	0	0,4870689655172413
P-3Q	7	3080	11160	10418	742	742	0	0,4695648631029986
3P+3Q	9	36198	142362	129264	13098	13098	0	0,46068172704972166
3P-3Q	11	436226	1841566	1629950	211616	211410	206	∼ 0,45538172416369477
	13	5362444	24102612	20823740	3278872	3267834	11038	∼ 0,451996025049843
3P+0	1	2	6	6	0	0	0	0,75
3P-Q	3	24	72	72	0	0	0	0,525
P+3Q	5	262	890	852	38	38	0	0,4870689655172413
P-3Q	7	3080	11160	10418	742	742	0	0,4695648631029986
3P+3Q	9	36198	142362	129264	13098	13098	0	0,46068172704972166
3P-3Q	11	436226	1841566	1629950	211616	211410	206	∼ 0,45538172416369477
	13	5362444	24102612	20823740	3278872	3267834	11038	∼ 0,451996025049843
0+3Q	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	236	916	832	84	84	0	0,4967672413793103
P+3Q	7	2668	11988	10456	1532	1532	0	0,4806611927398444
P-3Q	9	31660	160148	132224	27924	27924	0	0,4730955646473569
3P-3Q	11	388036	2174332	1714468	459864	454880	4984	∼ 0,46867988331021054
	13	4888112	29901200	22565020	7336180	7170144	166036	∼ 0,4660883492048925
0+3Q	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	236	916	832	84	84	0	0,4967672413793103
P+3Q	7	2668	11988	10456	1532	1532	0	0,4806611927398444
P-3Q	9	31660	160148	132224	27924	27924	0	0,4730955646473569
3P+3Q	11	388036	2174332	1714468	459864	454880	4984	∼ 0,46867988331021054
	13	4888112	29901200	22565020	7336180	7170144	166036	∼ 0,4660883492048925

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2, ∼ ≙ geschätzter Wert

Tabelle 21: (Fortsetzung)

FP	w	pos	neg	pos-1	neg-1	pos-2	neg-2	AJHD
0+3Q	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	262	890	852	38	38	0	0,4870689655172413
P+3Q	7	3080	11160	10418	742	742	0	0,4695648631029986
3P+3Q	9	36198	142362	129264	13098	13098	0	0,46068172704972166
3P-3Q	11	436226	1841566	1629950	211616	211410	206	$\sim$ 0,45538172416369477
	13	5362444	24102612	20823740	3278872	3267834	11038	$\sim$ 0,451996025049843
0+3Q	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
3P-Q	5	262	890	852	38	38	0	0,4870689655172413
P-3Q	7	3080	11160	10418	742	742	0	0,4695648631029986
3P+3Q	9	36198	142362	129264	13098	13098	0	0,46068172704972166
3P-3Q	11	436226	1841566	1629950	211616	211410	206	$\sim$ 0,45538172416369477
	13	5362444	24102612	20823740	3278872	3267834	11038	$\sim$ 0,451996025049843
0+3Q	1	2	6	6	0	0	0	0,75
3P+Q	3	24	72	72	0	0	0	0,525
P+3Q	5	246	906	864	42	42	0	0,48976293103448276
P-3Q	7	2834	11662	10672	990	990	0	0,47474556084885233
3P+3Q	9	33610	152982	134576	18406	18406	0	0,46684833430742256
3P-3Q	11	410360	2037352	1727002	310350	309300	1050	$\sim$ 0,4621660247657023
	13	5128440	27469192	22493872	4975320	4927890	47430	$\sim$ 0,45922627121476856
0+3Q	1	2	6	6	0	0	0	0,75
3P-Q	3	24	72	72	0	0	0	0,525
P+3Q	5	246	906	864	42	42	0	0,48976293103448276
P-3Q	7	2834	11662	10672	990	990	0	0,47474556084885233
3P+3Q	9	33610	152982	134576	18406	18406	0	0,46684833430742256
3P-3Q	11	410360	2037352	1727002	310350	309300	1050	$\sim$ 0,4621660247657023
	13	5128440	27469192	22493872	4975320	4927890	47430	$\sim$ 0,45922627121476856
3P+Q	1	2	6	6	0	0	0	0,75
3P-Q	3	24	72	72	0	0	0	0,525
P+3Q	5	216	936	848	88	88	0	0,49999999999999994
P-3Q	7	2872	12104	10912	1192	1192	0	0,4792383820998279
3P+3Q	9	36744	156920	137872	19048	19048	0	0,46885697104264973
3P-3Q	11	459696	2051024	1758280	292744	292744	0	0,46246004471356833
	13	5762024	27054360	22670352	4384008	4384008	0	0,4582485972285563

FP $\widehat{=}$ Fehlende Punkte bei der Rekodierung, $w \widehat{=}$ Fensterbreite in Bits, pos $\widehat{=}$ Anzahl der positiven Rekodierungen
neg $\widehat{=}$ Anzahl der negativen Rekodierungen, pos-1 $\widehat{=}$ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 $\widehat{=}$ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 $\widehat{=}$ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 $\widehat{=}$ Anzahl der negativen Rekodierungen mit Nullspalten-2, $\sim \widehat{=}$ geschätzter Wert

B.1. Zusätzliche Nullspalten

Tabelle 22: Ergebnisse der resultierenden AJHD des FW-Shamir Verfahrens mit vier vorzuberechnenden Punkten und zusätzlichen Nullspalten

FP	w	pos	pos+1	neg	pos-1	neg-1	pos-2	neg-2	AJHD
3P+0	1	2	0	6	6	0	0	0	0,75
0+3Q	3	24	0	72	72	0	0	0	0,525
3P+Q	5	284	0	868	856	12	12	0	0,48060344827586216
3P-Q	7	3380	0	10508	10292	216	216	0	0,46138100436681223
3P+3Q	9	40416	0	127712	124400	3312	3312	0	0,45109681577314176
3P-3Q	11	485352	0	1558040	1509752	48288	48288	0	0,44492055546474885
3P+0	1	2	0	6	6	0	0	0	0,75
0+3Q	3	24	0	72	72	0	0	0	0,525
P+3Q	5	284	0	868	856	12	12	0	0,48060344827586216
P-3Q	7	3380	0	10508	10292	216	216	0	0,46138100436681223
3P+3Q	9	40416	0	127712	124400	3312	3312	0	0,45109681577314176
3P-3Q	11	485352	0	1558040	1509752	48288	48288	0	0,44492055546474885
3P+Q	1	2	0	6	6	0	0	0	0,75
3P-Q	3	24	0	72	72	0	0	0	0,525
P+3Q	5	216	0	936	848	88	88	0	0,49999999999999994
P-3Q	7	2872	0	12104	10912	1192	1192	0	0,4792383820998279
3P+3Q	9	36712	32	156920	137872	19048	19048	0	0,4687966142852188
3P-3Q	11	459272	424	2051024	1758280	292744	292744	0	0,4623973078331728

FP ≙ Fehlende Punkte bei der Rekodierung, w ≙ Fensterbreite in Bits, pos ≙ Anzahl der positiven Rekodierungen
neg ≙ Anzahl der negativen Rekodierungen, pos-1 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-1
neg-1 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-1, pos-2 ≙ Anzahl der positiven Rekodierungen mit Nullspalten-2
neg-2 ≙ Anzahl der negativen Rekodierungen mit Nullspalten-2