RS - Übung 6

Arne Beer (MN 6489196),
Rafael Epplee (MN 6269560),
Julian Polatynski (MN 6424884)

December 5, 2012

6.1

Die gerade Anzahl der Codewörter erklärt sich auf dem Aufbau des Codes, bis auf das erste bit ist er nämlich Spiegelsymmetrisch:

0	00
0	01
0	11
0	10
1	10
1	11
1	01
1	00

So kann man jeweils das erste und das letzte Wort wegnehmen, da sich dann die neuen äußeren bits nur durch ihre ersten bits unterscheiden. Bei einer ungeraden Anzahl an Codewörtern würde es keine teilweise Symmetrie und damit auch keinen einschrittigen Code geben können.

6.2

a)

Die Minimaldistanz dieses Codes ist 4. Wenn man ein Datenbit ändert, so ändert sich damit zwangsweise auch

1. Das Paritätsbit für die Zeile
2. Das Paritätsbit für die Spalte
3. Und damit auch das Paritätsbit für die Paritätsbits der Spalten.

Insgesamt ändern sich durch die Änderung eines Datenbits also 4 Bits.

b)

Generell gilt nach der in der Vorlesung eingeführten Formel: Es sind $d - 1 = 3$-Bit-Fehler auffindbar. Außerdem gilt: es sind $\frac{d-1}{2} = 1$-Bit-Fehler korrigierbar.

Dies lässt sich wie folgt erklären: Einbitfehler sind durch die Paritätsbits erkennbar und genau lokalisierbar: bei einem Fehler in den Datenbits geben die zwei geänderten Paritätsbits die Position des Fehlers an; Bei einem Fehler in den Paritätsbits ist alles bis auf das Paritätsbit korrekt, wodurch man den Fehler auch direkt lokalisieren kann. Durch das exakte lokalisieren der Fehler lassen sie sich dann korrigieren.

Zwei- und Dreibitfehler können so angeordnet sein, dass eigentlich falsche Paritätsbits zu stimmen scheinen, oder dass zwei falsche Datenbits ihre Wirkung auf das Paritätsbit aufheben und trotzdem zu einem richtigen Paritätsbit führen. Diese Fehler sind immer erkennbar, da bei drei Änderungen immer entweder eines der drei Paritätsbits oder ein Datenbit nicht mit dem Rest übereinstimmt. Doch sie können nicht immer korrigiert werden, da die oben genannten Effekte dazu führen können, dass jedes Bit bis auf eines richtig erscheint, und ein Bit reicht nicht, um alle drei falschen Bits zu lokalisieren.

c)

Für einen Vierbitfehler, der nicht erkannt wird, könnte man $d_{0,0}$, $p_{8,0}$, $p_{0,8}$ und $p_{8,8}$ ändern. Dann würden beide Paritätsbits passen und auch das Paritätsbit über die Paritätsbits der Spalten wäre richtig. Somit wäre der Fehler nicht erkennbar.

d)

So einen Vierbitfehler gibt es für jedes Datenbit ein mal, also insgesamt 64 mal.

Die Anzahl aller möglichen Vierbitfehler errechnet sich aus jeder möglichen Teilmenge aller Bits, also

$$\binom{64+8+8+1}{4} = \binom{81}{4} = 1.663.740$$

Der Anteil dieser nicht erkennbaren Fehler an der Gesamtanzahl ist

$$\frac{64}{1.663.740} = \frac{16}{415.935} \approx 0,000038468$$

6.3

a)

Das verfälschte Codewort würde folgendermaßen aussehen: 0011000
Die dazugehörigen Prüfbits wären:$x_a = 1$, $x_b = 1$, $x_c = 1$
Man sieht, dass alle drei Teile des Prüfwortes des verfälschten Codeworts vom ursprünglichen

Codewort abweichen. Nun muss nach der Ziffer gesucht werden, die auf alle 3 Prüfbits Einfluss hat und somit eine Veränderung dieser Bits bewirken konnte. Zu sehen ist, dass nur die Stelle c_7 in allen 3 Prüfbits relevant ist. Daraus lässt sich schlussfolgern, dass c_7 fehlerhaft sein muss.

b)

Generatormatrix.

Die Generatormatrix G erstellen wir hier in mehreren Schritten. Zuallererst wird die Art des Codes bestimmt, was in dem vorliegenden Fall ein (31,26)-Hamming Code ist. Es gibt also 5 Prüfbits. Die Generatormatrix muss die Form 31x26 haben, sodass aus dem ursprünglichen 26-Bit Codewort das 31 Bit-Codewort mit den Prüfbits entsteht. Wir nehmen anfangs eine Null-Matrix und fügen dann in den folgenden Schritten die fehlenden Einsen hinzu.

Anhand eines Beispieles lassen sich die folgenden Schritte am besten darstellen. Man nehme das Zeichen d_i an der Position i und überlege sich, an welcher Stelle es in dem Hamming Codewort stehen würde und nenne diese Stelle j. Nun setzt man eine 1 an die Stelle j/i der Matrix, wobei hier nach dem üblichen Schema Zeile/Spalte geordnet wird. Alle nachfolgenden Zeilen dieser Spalte sind von nun an Nullen, da die Ziffer in den nachfolgenden Prüfbits nicht mehr erfasst wird.

Nun überlege man sich, welche Prüfziffern p_k das Zeichen d_i erfassen und setze an die Stelle k/i, die die Prüfziffer im Hamming-Code einnehmen würde, eine 1. Dies muss für alle Prüfziffern bis zur Stelle c_j des Hamming-Codes fortgesetzt werden. Die nun erhaltene Generatormatrix G erfüllt die Bedingung: $c = G \cdot d$.

Prüfmatrix.

Um die Prüfmatrix H zu erstellen benötigt man zuerst die Anzahl der Bits des Hamming-Codes und die Anzahl der Prüfbits. Im vorliegenden Fall haben wir 31-Bits mit 5-Prüfbits. Folglich benötigen wir eine 5x31 Matrix. Wie zuvor wählen wir zuerst eine Null-Matrix.

Die Spalten repräsentieren die jeweilige Stelle des Codes und die Zeilen den jeweiligen Prüfbit. Nehmen wir nun erneut das Beispiel das die Zeile m das Erfassungsbild des Prüfbits p_m repräsentiert, jedoch inklusive des Prüfbits selbst.

Wenn für das Prüfbit p_m im Hammingcode eine Ziffer c_n erfasst wird, muss an der Stelle m/n in der Matrix eine 1 gesetzt werden. Es ist darauf zu achten, dass auch an der Stelle des Prüfbits selbst ein Prüfbit gesetzt wird.

Da durch die Modulo 2 Eigenschaft des Prüfbits gewährleistet ist, dass die Summe des Prüfbits und aller seiner geprüften Bits immer eine gerade Zahl ist, muss die Prüfmatrix, sobald jede Stelle modulo 2 genommen wurde eine 3x1 0-Matrix ergeben, solange der Code nicht fehlerhaft ist.