Inleiding Informatietheorie

(SV 1.1)

P.J. den Brok MA

3 januari 2005

Inhoudsopgave

Hoofdstuk 1

Informatie

In het spraakgebruik wordt het woord '*informatie*' vaak gebruikt voor begrippen zoals '*gegevens*', '*kennis*', '*ideeën*' en '*visie*'.

> *"Before you become too entranced with gorgeous gadgets and mesmerizing video displays, let me remind you that information is not knowledge, knowledge is not wisdom, and wisdom is not foresight. Each grows out of the other, and we need them all."* (Arthur C. Clarke).

Grootheden met een subjectieve betekenis zoals 'kennis', 'ideeën' en 'visies' zijn moeilijk te beschrijven. Daarentegen blijkt het redelijk eenvoudig het begrip 'informatie' in operationele termen te beschrijven. Een bruikbare definitie van informatie is de eerste stap geweest bij de ontwikkeling van de informatietheorie. Deze informatietheorie, een peiler voor de huidige informatiemaatschappij, is grotendeels het werk geweest van Claude E. Shannon.

1.1 De informatie over een gebeurtenis of selectieve informatie

Intuïtief wordt de informatie over een gebeurtenis hoger gewaardeerd naarmate zij een groter verassingseffect heeft.

Voorbeeld 1.1

1 "*Morgen is er weer een dag.*"
2 "*Op 1 april worden alle belastingen afgeschaft.*"

Boodschap 1 is zeer waarschijnlijk, zij geeft ons niet veel informatie. Boodschap 2 is daarentegen zeer onwaarschijnlijk, zij verrast ons het meest en bevat daarom meer informatie dan boodschap 1.

Voorbeeld 1.2

In het taalgebruik worden onwaarschijnlijke uitspraken *sterke uitspraken* genoemd. Uitspraken die voor de hand liggen, worden *zwakke uitspraken* genoemd. Sterke uitspraken hebben een hogere informatieve waarde dan zwakke uitspraken. Beursprofeten, handlijnenlezers, koffiedikkijkers en horoscooptrekkers zijn experts in het maken van zwakke uitspraken, zoals "*De aandelen gaan zakken.*" en "*U gaat op reis.*". Op deze manier kan de inhoudelijke informatieve waarde van de boodschap amper bepaald worden.

Voor een objectieve waarnemer is het verassingsvermogen van een gebeurtenis gemakkelijker te bepalen dan de interesse bij de ontvanger van de boodschap. Daarom zal de niet-intuïtieve definitie van informatie $H(X = x_i)$ zich beperken tot de omgekeerde waarschijnlijkheid, de *onwaarschijnlijkheid* $1/p_i$ van de gebeurtenis $X = x_i$. Bij het opstellen van de definitie van informatie moet men ook rekening houden met de intuïtieve eigenschap dat de informatie over een samenstelling van onafhankelijke gebeurtenissen $X = x_i, Y = y_j$ gelijk is aan de som van de afzonderlijke componenten $X = x_i$ en $Y = y_j$:

$$H(X = x_i, Y = y_j) = H(X = x_i) + H(Y = y_j)$$

Voordat de definitie van informatie wordt gegeven, moet eerst de formele notatie nader worden verklaard. De notatie $H(X = x_i)$ geeft de selectieve informatie van een *variabele* of een *stochast* X met een *uitkomst* x_i. De variabele X heeft een verzameling $\{x_1, x_2, x_3, \ldots, x_n\}$ van $|X| = n$ uitkomsten. Deze uitkomsten noemen wij soms gebeurtenissen, karakters, symbolen of letters. Op deze uitkomstenverzameling is een kansfunctie $P(X = x_i) = p_i$ gedefinieerd, die aan elke uitkomst x_i een kanswaarde p_i toekent. De *selectieve informatie* $H(X = x_i)$ over een uitkomst x_i met een kans $0 < P(X = x_i) = p_i < 1$, is gedefinieerd als de logaritme met het grondtal 2 ($log_2 x = ln(x)/ln(2) = \mathrm{ld}\,(x)$) van de onwaarschijnlijkheid $1/p_i$, van deze uitkomst:

$$H(X = x_i) = \mathrm{ld}\,(\tfrac{1}{p_i}) = -\mathrm{ld}\,(p_i) \qquad \text{[bit]} \tag{1.1}$$

Aan deze definitie van de selectieve informatie is de eenheid *bit* gekoppeld. De selectieve informatie is met deze eenheid gelijk aan het gemiddelde aantal ja/nee antwoorden dat nodig is om de betreffende uitkomst vast te stellen.

Voorbeeld 1.3

De selectieve informatie over het werpen van 4 ogen met een ideale dobbelsteen is $H(X = 4) = \mathrm{ld}\,(6) = 2,58$ bit. Hieruit blijkt dat informatie niet altijd geheeltallig is. De informatiewaarde in bit komt immers overeen met het *gemiddelde* aantal ja/nee antwoorden.

Zoals al eerder gesteld is, moet de definitie van de selectieve informatie voldoen aan de intuïtieve wens dat de selectieve informatie over een complex van onafhankelijke gebeurtenissen de som is van de informatie van de samenstellende gebeurtenissen. De definitie van selectieve informatie (formule: 1.1) voldoet aan deze wens:

$$H(X=x,Y=y) = \mathrm{ld}\,(\tfrac{1}{p_x}\cdot\tfrac{1}{p_y}) = \mathrm{ld}\,(\tfrac{1}{p_x}) + \mathrm{ld}\,(\tfrac{1}{p_y}) = H(X=x)+H(Y=y) \quad \text{[bit]} \quad (1.2)$$

Voorbeeld 1.4

> Om één van de 64 vakjes op een schaakbord aan te geven, zijn gemiddeld $H(X = x,Y = y) = \mathrm{ld}\,(8)+\mathrm{ld}\,(8) = \mathrm{ld}\,(64) = 6$ ja/nee antwoorden noodzakelijk. De 6 bit informatie kan men verdelen in 3 bit informatie over de x-positie en 3 bit informatie over de y-positie. De gezamenlijke informatie over de x- en de y-positie is de som van beide.

In het algemeen heeft de informatie over een uitkomst een positieve reële waarde, met uitzondering van een onmogelijke- of een onvermijdelijke uitkomst. Hoewel informatie over een onmogelijke gebeurtenis zinvol kan zijn voor liefhebbers van science-fictionverhalen, wordt dit in het algemeen als verspilling van tijd en energie beschouwd. Daarom is op formele gronden de informatie over een *onmogelijke uitkomst* ongedefinieerd:

$$\lim_{p\downarrow 0}(-\mathrm{ld}\,(p)) = \infty \quad (1.3)$$

Een *onvermijdelijke uitkomst* geeft geen verhoging van de informatie. Aan de informatie over een onvermijdelijke uitkomst wordt om deze formele reden de nulwaarde toegekend:

$$\lim_{p\uparrow 1}(-\mathrm{ld}\,(p)) = 0 \quad (1.4)$$

1.2 De gemiddelde informatie of entropie

Technische toepassingen van de informatietheorie maken meestal gebruik van een andere interpretatie van informatie. Om deze interpretatie te onderscheiden van het begrip selectieve informatie, introduceren wij het begrip *gemiddelde informatie* of *entropie*. Entropie staat voor begrippen zoals *chaos*, *onvoorspelbaarheid* of *onzekerheid*. Wat heeft onzekerheid met informatie te maken?

> **Informatie is de mate van afname van de onzekerheid bij de ontvanger van de boodschap.**

Zowel zeer waarschijnlijke als zeer onwaarschijnlijke gebeurtenissen zijn naar ons gevoel tamelijk voorspelbaar. Een zeer waarschijnlijke gebeurtenis treedt bijna altijd op, een zeer onwaarschijnlijke gebeurtenis treedt bijna nooit op. De grootste onzekerheid hebben wij echter bij de minder extreme gebeurtenissen. Indien wij de gemiddelde informatie van alle $|X|$ uitkomsten nemen, zijn het vooral de minder extreme gebeurtenissen die het gemiddelde en de onzekerheid bepalen. De entropie $H(X)$ van een variabele X wordt daarom formeel gedefinieerd als:

$$H(X) = E(H(X = x_i)) = -\sum_{i=1}^{|X|} p_i \cdot \mathrm{ld}\,(p_i) \qquad \text{[bit]} \tag{1.5}$$

Ook de grootheid entropie heeft de eenheid [bit]. De entropie komt dan overeen met het gemiddelde aantal ja/nee antwoorden dat nodig is om een willekeurige uitkomst te bepalen.

Waarom zijn wij geïnteresseerd in de entropie? Voor toepassingen zoals compressie, communicatie en cryptografie zijn categorieën van berichten of boodschappen belangrijker dan individuele gebeurtenissen en boodschappen. Daarnaast is het nog steeds mogelijk de entropie van samenstellingen te berekenen aan de hand van de entropieën van de afzonderlijke variabelen. Bijvoorbeeld, de entropie of de gemiddelde informatie van een rij van L *onafhankelijke variabelen* X is gelijk aan:

$$H(X^L) = H(\overbrace{X, X, \ldots, X}^{L}) = L \cdot H(X) \qquad \text{[bit]} \tag{1.6}$$

De waarde $L \cdot H(X)$ is het gemiddelde van de som van L steekproeven met teruglegging. Dit resultaat is niet geldig als steekproeven afhankelijk van elkaar zijn. Voordat wij de relatie tussen entropie en samenstellingen van afhankelijke variabelen behandelen, analyseren wij eerst de invloed van de kansverdeling op de entropie:

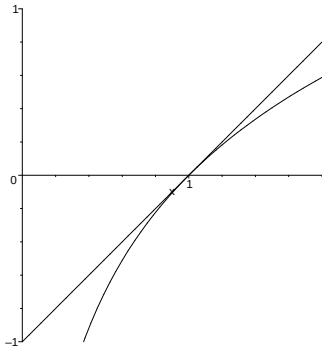
De minimale entropie: De entropie van een variabele wordt nul, indien een uitkomst x met 100% waarschijnlijkheid plaats zal vinden. Het is vanzelfsprekend dat er geen andere uitkomsten mogelijk zijn:

$$H(X) = -\lim_{p \uparrow 1} (p \cdot \mathrm{ld}\,(p)) = 0 \tag{1.7}$$

Dit geldt ook als een uitkomst met 0% waarschijnlijkheid, dus met 100% zekerheid niet plaats zal vinden:

$$H(X) = -\lim_{p \downarrow 0} (p \cdot \mathrm{ld}\,(p)) = 0 \tag{1.8}$$

De maximale entropie: De maximale entropie $H_{max}(X)$ van een variabele X is de entropie die zij zou hebben als alle elementaire uitkomsten even waarschijnlijk zouden zijn. De maximale entropie $H_{max}(X)$ is dus alleen afhankelijk van het aantal elementaire uitkomsten $|X|$ van die variabele X. Met andere woorden, de maximale entropie treedt op als de verzameling uitkomsten van X een 'uniforme verdeling' zou hebben.



Figuur 1.1: $\ln(x) \leq x-1$

Omdat de ongelijkheid $\ln(x) \leq x-1$ geldig is voor alle waarden van x (zie figuur 1.1) kunnen wij haar gebruiken bij het bepalen van de maximale entropie $H_{max}(X)$:

$$\begin{aligned}
H(X) - \mathrm{ld}(|X|) &= E(-\mathrm{ld}(P(X))) - \mathrm{ld}(|X|) \\
&= E(-\mathrm{ld}(P(X)) - \mathrm{ld}(|X|)) && (E(X)+\alpha = E(X+\alpha)) \\
&= \tfrac{1}{\ln 2} E(\ln \tfrac{1}{|X|\cdot P(X)}) \\
&\leq \tfrac{1}{\ln 2} E(\tfrac{1}{|X|\cdot P(X)} - 1) && (\ln \tfrac{1}{|X|\cdot P(X)} \leq \tfrac{1}{|X|\cdot P(X)} - 1) \\
&= \tfrac{1}{\ln 2} \textstyle\sum_{i=1}^{|X|} p_i \left(\tfrac{1}{|X|\cdot p_i} - 1\right) \\
&= \tfrac{1}{\ln 2} \textstyle\sum_{i=1}^{|X|} \left(\tfrac{1}{|X|} - p_i\right)
\end{aligned}$$

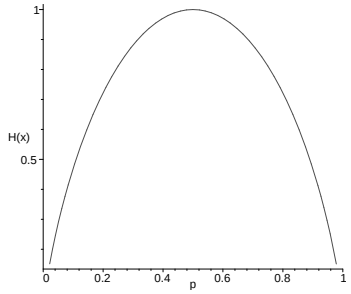
Deze laatste uitdrukking wordt 0 als $p_i = \frac{1}{|X|}$:

$$H(X) - \mathrm{ld}(|X|) \leq 0 \qquad \Rightarrow \qquad H(X) \leq \mathrm{ld}(|X|)$$

De entropie van een variabele X met $|X|$ elementaire uitkomsten is maximaal als zij een uniforme verdeling heeft. De maximale entropie is in dat geval:

$$H_{max}(X) = \mathrm{ld}(|X|) \qquad \text{[bit]} \tag{1.9}$$

Voorbeeld 1.5



Figuur 1.2: De entropie van een Bernoulli variabele

Een *Bernoulli variabele X* heeft een uitkomstenverzameling $\{x_1, x_2\}$ met de kansen $\{p_1 = p, p_2 = 1-p\}$. De entropie van deze variabele wordt bepaald met:

$$H(X) = -p \cdot \mathrm{ld}\,(p) - (1-p) \cdot \mathrm{ld}\,(1-p)$$

Uit figuur 1.2 blijkt dat de entropie van een Bernoulli variabele het grootst is als beide uitkomsten even waarschijnlijk zijn. Deze figuur is te interpreteren als de relatie tussen waarschijnlijkheid (de x-as) en de onzekerheid (de y-as) van het werpen van een muntstuk. Indien het muntstuk een voorkeur heeft voor de 'kop', dan is de onzekerheid van de uitkomst minder, idem dito voor de 'munt'. De onzekerheid verdwijnt indien de munt altijd op 'kop' of altijd op 'munt' valt. Een gokker zal bij volledige onzekerheid, de fifty/fifty situatie aannemen.

1.3 De entropie van samengestelde variabelen

Boodschappen, berichten, talen en codes kunnen vaak beschouwd worden als samenstellingen van variabelen. De entropie van een bericht wordt opgebouwd uit de entropie van de afzonderlijke variabelen en hun onderlinge relaties. Deze variabelen kunnen meer of minder afhankelijk van elkaar zijn, zoals de lettercombinaties in natuurlijke woorden. Het is niet noodzakelijk dat variabelen dezelfde uitkomstenverzameling delen, of dezelfde kansfunctie bezitten. Wij zullen aan de hand van de meest eenvoudige samenstelling, een tweetal variabelen X en Y, de resulterende entropie analyseren. De samengestelde verzameling uitkomsten wordt in de volgende matrix aangegeven:

$$\left\{ \begin{array}{cccc} x_1,y_1 & x_1,y_2 & \dots & x_1,y_m \\ x_2,y_1 & x_2,y_2 & \dots & x_2,y_m \\ \vdots & \vdots & \vdots & \vdots \\ x_n,y_1 & x_n,y_2 & \dots & x_n,y_m \end{array} \right\}$$

Op deze uitkomstenverzameling is een kansfunctie $P(X = x_i, Y = y_j) = p_{i,j}$ gedefinieerd:

$$P(X,Y) = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,m} \end{pmatrix}$$

De samengestelde entropie $H(X,Y)$ wordt gedefinieerd als:

$$H(X,Y) = E(H(X = x_i, Y = y_j)) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{i,j})\right) \tag{1.10}$$

Wij zullen nu de invloed van afhankelijkheid op de entropie van de samengestelde variabelen onderzoeken. Wij beginnen met de definitie van de *selectieve entropie* van een variabele Y indien de uitkomst $X = x_i$ gegeven is:

$$H(Y|X = x_i) = -\sum_{j}^{|Y|}\left(\frac{p_{i,j}}{p_i} \cdot \mathrm{ld}\,(\tfrac{p_{i,j}}{p_i})\right) = -\sum_{j}^{|Y|}\left(p_{j|i} \cdot \mathrm{ld}\,(p_{j|i})\right) \tag{1.11}$$

Deze selectieve entropie is een functie van de vrije variabele *i* en wordt beschouwd als een variabele $Y|X$. De entropie van deze variabele is de hoeveelheid entropie die de variabele Y bijdraagt aan de totale entropie. Wij noemen deze hoeveelheid *conditionele entropie* of *equivocatie* $H(Y|X)$:

$$H(Y|X) = E(H(Y|X = x_i)) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_i \cdot (p_{j|i} \cdot \mathrm{ld}\,(p_{j|i})\right) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{j|i})\right) \tag{1.12}$$

Wat kunnen wij zeggen over de entropie van de samengestelde variabelen indien de variabelen Y onafhankelijk is van de variabele X? Als zij onafhankelijk zijn, geldt dat $p_{j|i} = p_j$ en $p_{i,j} = p_i \cdot p_j$ Hiermee kunnen wij het volgende resultaat afleiden:

$$H(Y|X) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{j|i})\right) = \sum_{i}^{|X|} p_i \cdot \left(-\sum_{j}^{|Y|}\left(p_j \cdot \mathrm{ld}\,(p_j)\right)\right) = \left(\sum_{i}^{|X|} p_i\right) \cdot H(Y) = H(Y)$$

Als variabele Y onafhankelijk is van variabele X dan geldt voor de equivocatie:

$$H(Y|X) = H(Y) \tag{1.13}$$

Dit is tevens de grootste waarde die $H(Y|X)$ kan aannemen. Indien variabele Y wel afhankelijk is van variabele X dan geldt voor de equivocatie:

$$H(Y|X) \leq H(Y) \tag{1.14}$$

Deze equivocatie wordt zelfs 0 als de gebeurtenis Y volledig afhankelijk is van de gebeurtenis X. Bij volledige afhankelijkheid geldt voor alle y_j en x_i dat $p_{j|i} = 0$ of $p_{j|i} = 1$. Uit de vergelijkingen 1.8 en 1.7 kan men concluderen dat het product $p_{j|i} \cdot \mathrm{ld}\,(p_{j|i})$ altijd 0 is:

$$H(Y|X) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{j|i})\right) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_i \cdot \overbrace{p_{j|i} \cdot \mathrm{ld}\,(p_{j|i})}^{0}\right) = 0 \tag{1.15}$$

Wij kunnen dus concluderen dat een samengestelling van onafhankelijke variabelen meer entropie bevat dan een samenstelling van afhankelijke variabelen. Als bovendien de uitkomsten van de samenstellende variabelen uniform verdeeld zijn, is de entropie maximaal.

Wat is nu de relatie tussen de samengestelde entropie $H(X,Y)$ en $H(Y|X)$?

$$H(X,Y) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{j|i} \cdot p_i)\right) = -\sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_i)\right) - \sum_{i}^{|X|}\sum_{j}^{|Y|}\left(p_{i,j} \cdot \mathrm{ld}\,(p_{j|i})\right) \tag{1.16}$$

Hieruit blijkt dat:

$$H(X,Y) = H(X) + H(Y|X) \tag{1.17}$$

Op dezelfde manier kunnen wij bewijzen dat $H(Y,X) = H(Y) + H(X|Y)$. De relatie tussen entropie, gemeenschappelijke entropie en equivocatie wordt als volgt samengevat:

$$H(X,Y) = H(Y) + H(X|Y) = H(X) + H(Y|X) \leq H(X) + H(Y) \tag{1.18}$$

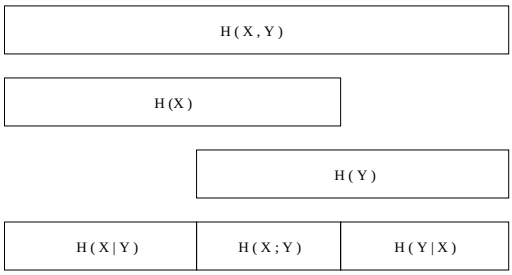
De laatste ongelijkheid wordt een gelijkheid als X en Y onafhankelijk van elkaar zijn. Om deze afhankelijkheid te kwantificeren, introduceren wij het begrip *onderlinge entropie* $H(X;Y)$:

$$H(X;Y) = H(X) + H(Y) - H(X,Y) \tag{1.19}$$

Met behulp van formule 1.18 kunnen wij afleiden:

$$H(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{1.20}$$

Dit valt op een gemakkelijke manier te zien in figuur 1.3:



Figuur 1.3: Onderlinge entropie

Als X en Y volledig onafhankelijk van elkaar zijn, dan is de onderlinge entropie $H(X;Y) = 0$. Indien X en Y volledig afhankelijk van elkaar zijn, dan is de onderlinge entropie gelijk aan beide entropieën $H(X;Y) = H(X) = H(Y)$

Voorbeeld 1.6

Een eenvoudige taal bestaat uit woorden (samenstellingen) van 2 variabelen (letters). De letters van de woorden worden gekozen uit het *alfabet* (gemeenschappelijke uitkomsten): $\{a,b\}$. De relatieve frequenties $P(X,Y)$ van de 2-letter woorden zijn:

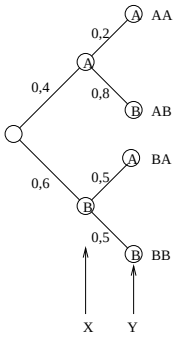
$P(X,Y)$	$x_1 = a$	$x_2 = b$	$P(Y)$
$y_1 = a$	$P(X,Y = aa) = 0,08$	$P(X,Y = ba) = 0,30$	$P(Y = a) = 0,38$
$y_2 = b$	$P(X,Y = ab) = 0,32$	$P(X,Y = bb) = 0,30$	$P(Y = b) = 0,62$
$P(X = x)$	$P(X = a) = 0,40$	$P(X = b) = 0,60$	$1,00$

Bepaal de entropieën $H(X)$, $H(Y)$, $H(X,Y)$, $H(Y|X)$ en $H(X;Y)$.

Antwoorden

1. De entropie $H(X)$ wordt berekend uit de marginale kansverdeling $P(X)$. Hieruit volgt $H(X) = -0,40 \cdot \mathrm{ld}\,(0,40) - 0,60 \cdot \mathrm{ld}\,(0,60) = 0,971$ bit;
2. De entropie $H(Y)$ wordt berekend uit de marginale kansverdeling $P(Y)$. Hieruit volgt $H(Y) = -0,38 \cdot \mathrm{ld}\,(0,38) - 0,62 \cdot \mathrm{ld}\,(0,62) = 0,958$ bit;
3. De gemeenschappelijke entropie $H(X,Y)$ wordt berekend uit de kansverdeling $P(X,Y)$. Hieruit volgt dat $H(Y,X) = -0,08 \cdot \mathrm{ld}\,(0,08) - 0,32 \cdot \mathrm{ld}\,(0,32) - 0,3 \cdot \mathrm{ld}\,(0,3) - 0,3 \cdot \mathrm{ld}\,(0,3) = 1,860$ bit;
4. Uit de kansverdeling $P(X,Y)$ kunnen wij de conditionele kansverdeling $P(Y|X)$ bepalen met de formule van *Bayes* $P(Y|X) = P(X,Y)/P(X)$. Dit geeft ons de volgende tabel:

$P(Y\|X)$	$x_1 = a$	$x_2 = b$
$y_1 = a$	$P(Y = a\|X = a) = 0,20$	$P(Y = a\|X = b) = 0,50$
$y_2 = b$	$P(Y = b\|X = a) = 0,80$	$P(Y = b\|X = b) = 0,50$



Figuur 1.4: Grafische weergave van $P(Y|X)$

Deze tabel wordt grafisch weergegeven in figuur 1.4.

Uit de conditionele kansen $P(Y|X=a)$ en $P(Y|X=b)$ volgen de selectieve entropieën $H(Y|X=a) = -0,2 \cdot \mathrm{ld}\,(0,2) - 0,8 \cdot \mathrm{ld}\,(0,8) = 0,722$ bit en $H(Y|X=b) = -0,5 \cdot \mathrm{ld}\,(0,5) - 0,5 \cdot \mathrm{ld}\,(0,5) = 1,000$ bit. Uit de selectieve entropieën berekenen wij de equivocatie $H(Y|X) = P(X=a) \cdot H(Y|X=a) + P(X=b) \cdot H(Y|X=b) = 0,4 \cdot 0,722 + 0,6 \cdot 1 = 0,889$ bit. Ter controle kunnen wij de equivocatie $H(Y|X)$ toetsen met: $H(Y|X) = H(X,Y) - H(X) = 1,860 - 0,971 = 0,889$ bit.

5. De onderlinge entropie $H(X;Y) = H(X) + H(Y) - H(X,Y) = 0,971 + 0,958 - 1,860 = 0,069$ bit.

1.4 Markov-rijen

In het vorige voorbeeld bleek dat een variabele Y afhankelijk kon zijn van zijn directe voorganger X. Een samenstelling van variabelen waarin alle variabelen X_n afhankelijk zijn van hun directe (bestaande) voorganger X_{n-1} noemt men een *Markov-rij* van de eerste orde. Dit geldt niet voor de eerste variabele X_1 omdat de voorganger X_0 niet bestaat. Bij een Markov-rij van de eerste orde wordt de conditionele kansmatrix $P(Y|X)$ aangegeven als $P(X_n|X_{n-1})$. Er er sprake van een soort geheugenwerking.

Deze geheugenwerking ontbreekt als alle variabelen onafhankelijk zouden zijn van hun voorgangers. In dat geval is er sprake van een Markov-rij van de nulde orde: $P(X_n|X_{n-1}) = P(X_n)$.

Indien een willekeurig variabele afhankelijk is van k (bestaande) voorgangers, dan noemt men dat een Markov-rij van de k-de orde $P(X_n|X_{n-1}, X_{n-2}, \ldots, X_{n-k})$. Bij natuurlijke talen is er sprake van Markov-rijen van een hogere orde dan 2 vanwege de spellingsregels. Sommige Nederlandse woorden bevatten vaste combinaties van 3 letters zoals de “sch”. Bovendien geven grammaticale regels aanleiding tot geheugenwerking over zeer lange deelrijen.

Voorbeeld 1.7

Gegeven een bericht van 120 karakters uit een alfabet met de symbolen $\{A,B,C\}$.

$$
\begin{array}{cccccccccccccccccccc}
C & A & B & A & B & A & C & A & C & C & A & C & C & A & C & A & B & C & C & C \\
A & C & C & A & C & C & A & C & C & A & C & C & C & A & C & C & A & C & C & C \\
A & C & C & A & C & C & A & C & C & A & B & A & C & A & C & C & A & C & C & A \\
C & C & C & A & C & C & A & C & C & C & A & C & C & A & C & A & B & B & B & A \\
B & A & C & A & C & C & A & C & C & C & A & C & C & A & C & A & B & C & C & A \\
C & C & C & A & C & C & A & C & A & B & C & C & A & C & C & A & C & C & A & C \\
C & & & & & & & & & & & & & & & & & & & \\
\end{array}
$$

Wij tellen alle 120 zelfstandige letters, behalve het 121-ste karakter. Daarna worden alle 120 samenstellingen van twee letters geteld, inclusief het 121-ste karakter omdat er anders geen 120 tweetallen te vormen zijn. Tijdens het tellen worden combinaties zoals “BBB” als twee keer voor “BB” geteld. De resultaten staan in de volgende tabel:

A	B	C	AA	AB	AC	BA	BB	BC	CA	CB	CC
40	10	70	0	8	32	5	2	3	34	0	35

Uit deze tellingen volgen de respectievelijke kansen:

p_A	p_B	p_C	p_{AA}	p_{AB}	p_{AC}	p_{BA}	p_{BB}	p_{BC}	p_{CA}	p_{CB}	p_{CC}
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{7}{12}$	0	$\frac{1}{15}$	$\frac{4}{15}$	$\frac{1}{24}$	$\frac{1}{60}$	$\frac{1}{40}$	$\frac{17}{60}$	0	$\frac{7}{24}$

Uit deze kansen kunnen wij met behulp van de formule $p_{Y|X} = \frac{p_{XY}}{p_X}$ het volgende resultaat bereken:

$p_{A\|A}$	$p_{B\|A}$	$p_{C\|A}$	$p_{A\|B}$	$p_{B\|B}$	$p_{C\|B}$	$p_{A\|C}$	$p_{B\|C}$	$p_{C\|C}$
0	$\frac{1}{5}$	$\frac{4}{5}$	$\frac{1}{2}$	$\frac{1}{5}$	$\frac{3}{10}$	$\frac{1}{2}$	0	$\frac{1}{2}$

Beschouwen wij X als de eerste letter en Y als de tweede letter van een tweetal, dan kunnen wij de volgende entropieën berekenen:

1. De entropie van de zelfstandige variabelen $H(X_n)$ is gelijk aan:

$$
\begin{aligned}
H(X_n) = \textstyle\sum_i p_i \cdot \mathrm{ld}\,(p_i) &= -p_A \cdot \mathrm{ld}\,(p_A) - p_B \cdot \mathrm{ld}\,(p_B) - p_C \cdot \mathrm{ld}\,(p_C) \\
&= 1,28 \text{ bit}
\end{aligned}
$$

2. De conditionele entropie $H(X_n|X_{n-1})$ volgt uit:

$$
\begin{aligned}
H(X_n|X_{n-1}) &= \textstyle\sum_i \sum_j p_{i,j} \cdot \mathrm{ld}\,(p_{i|j}) \\
&= -p_{AA} \cdot \mathrm{ld}\,(p_{A|A}) - p_{AB} \cdot \mathrm{ld}\,(p_{B|A}) - p_{AC} \cdot \mathrm{ld}\,(p_{C|A}) \\
&\quad -p_{BA} \cdot \mathrm{ld}\,(p_{A|B}) - p_{BB} \cdot \mathrm{ld}\,(p_{B|B}) - p_{BC} \cdot \mathrm{ld}\,(p_{C|B}) \\
&\quad -p_{CA} \cdot \mathrm{ld}\,(p_{C|A}) - p_{CB} \cdot \mathrm{ld}\,(p_{B|C}) - p_{CC} \cdot \mathrm{ld}\,(p_{C|C}) \\
&= 0,95 \text{ bit}
\end{aligned}
$$

3. De entropie van alle tweetallen variabelen $H(X_{n-1},X_n)$ is gelijk aan:

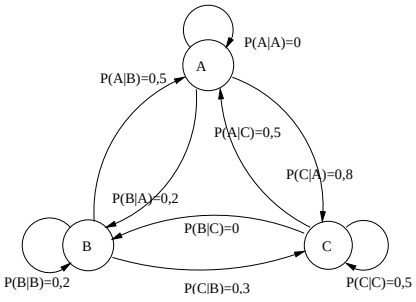
$$
\begin{aligned}
H(X_{n-1},X_n) &= \textstyle\sum_i \sum_j p_{i,j} \cdot \mathrm{ld}\,(p_{i,j}) \\
&= -p_{AA} \cdot \mathrm{ld}\,(p_{AA}) - p_{AB} \cdot \mathrm{ld}\,(p_{AB}) - p_{AC} \cdot \mathrm{ld}\,(p_{AC}) \\
&\quad -p_{BA} \cdot \mathrm{ld}\,(p_{BA}) - p_{BB} \cdot \mathrm{ld}\,(p_{BB}) - p_{BC} \cdot \mathrm{ld}\,(p_{BC}) \\
&\quad -p_{CA} \cdot \mathrm{ld}\,(p_{CA}) - p_{CB} \cdot \mathrm{ld}\,(p_{CB}) - p_{CC} \cdot \mathrm{ld}\,(p_{CC}) \\
&= 2,23 \text{ bit}
\end{aligned}
$$

Ook in dit voorbeeld blijkt dat $H(X_{n-1}, X_n) = H(X_n|X_{n-1}) + H(X_{n-1})$.

Wat is nu de entropie van het gehele bericht? Als alle karakters onafhankelijk van elkaar zouden zijn, dan is de totale entropie van het bericht gelijk aan $120 \cdot H(X_n) = 153,6$ bit. Wordt de eerste-orde afhankelijkheid van de variabelen meegewogen dan is de totale (gemiddelde) entropie lager dan bij onafhankelijke variabelen. Zij bedraagt $H(X_1) + 119 \cdot H(X_n|X_{n-1}) = 113,14$ bit. Hogere orde afhankelijkheid kan deze waarde nog verder verlagen.

Voorbeeld 1.8

Gegeven een oneindige samenstelling van variabelen. Elke variabele heeft dezelfde verzameling uitkomsten $\{A, B, C\}$. Bovendien is elke variabele afhankelijk van de vorige variabele. De gegeven overgangskansen zijn als graaf getekend, zie figuur 1.5. De knopen in de graaf zijn toestanden waarin de variabelen kunnen verkeren. De overgangskansen tussen deze toestanden worden weergegeven als gerichte takken. Per toestand wordt een vergelijking opgesteld, waarbij de *binnenkomende takken* de kans op deze toestand bepalen. De laatste vergelijking geeft aan dat alle toestanden de volledige verzameling van uitkomsten vormen. Bepaal de 0-de en de 1-orde entropie van een variabele.



Figuur 1.5: Toestandsdiagram van een eerste-orde Markov-rij

$$\begin{aligned} p_A &= p_{A|A} \cdot p_A + p_{A|B} \cdot p_B + p_{A|C} \cdot p_C \\ p_B &= p_{B|A} \cdot p_A + p_{B|B} \cdot p_B + p_{B|C} \cdot p_C \\ p_C &= p_{C|A} \cdot p_A + p_{C|B} \cdot p_B + p_{C|C} \cdot p_C \\ 1 &= p_A + p_B + p_C \end{aligned}$$

De gegeven overgangskansen zijn als coëfficiënten in het stelsel vergelijkingen opgenomen:

$$\begin{aligned} 0 &= -1 \cdot p_A + 1/2 \cdot p_B + 1/2 \cdot p_C \\ 0 &= 1/5 \cdot p_A - 4/5 \cdot p_B \\ 0 &= 4/5 \cdot p_A + 3/10 \cdot p_B - 1/2 \cdot p_C \\ 1 &= p_A + p_B + p_C \end{aligned}$$

Als wij dit stelsel vergelijkingen oplossen (zie: bijlage A) dan blijkt dat $p_A = \frac{1}{3}$, $p_B = \frac{1}{12}$ en $p_C = \frac{7}{12}$. Hieruit volgt dat de 0-orde entropie $H_0(X) = H(X_n)$ van een willekeurige variabele gelijk is aan $1,28$ bit.

Om de 1-ste orde entropie $H_1(X) = H(X_n|X_{n-1})$ te berekenen, moeten wij eerst de selectieve entropie $H(X_n|X_{n-1} = x_i)$ bepalen. Dit is mogelijk door per toestand de selectieve entropie van de *uitgaande takken* te berekenen met formule 1.11:

$$\begin{array}{lcl} H(X_n|X_{n-1} = \text{'A'}) & = & -p_{A|A} \cdot \mathrm{ld}\,(p_{A|A}) - p_{B|A} \cdot \mathrm{ld}\,(p_{B|A}) - p_{C|A} \cdot \mathrm{ld}\,(p_{C|A}) \;=\; 0,72 \\ H(X_n|X_{n-1} = \text{'B'}) & = & -p_{A|B} \cdot \mathrm{ld}\,(p_{A|B}) - p_{B|B} \cdot \mathrm{ld}\,(p_{B|B}) - p_{C|B} \cdot \mathrm{ld}\,(p_{C|B}) \;=\; 1,48 \\ H(X_n|X_{n-1} = \text{'C'}) & = & -p_{A|C} \cdot \mathrm{ld}\,(p_{A|C}) - p_{B|C} \cdot \mathrm{ld}\,(p_{B|C}) - p_{C|C} \cdot \mathrm{ld}\,(p_{C|C}) \;=\; 1,00 \end{array}$$

De 1-orde entropie $H_1(X) = H(X_n|X_{n-1})$ van een willekeurige variabele wordt verkregen door de selectieve entropieën te middelen over alle toestanden (formule 1.12):

$$H_1(X) = H(X_n|X_{n-1}) = \frac{1}{3} \cdot 0,72 + \frac{1}{12} \cdot 1,48 + \frac{7}{12} \cdot 1,00 = 0,95 \text{ bit}$$

1.5 Redundantie

Wij hebben gezien dat de hogere-orde entropie vaak lager is dan de maximale entropie $H_{max}(X)$. Indien wij alle mogelijke afhankelijkheden in achtnemen, nadert de entropie een ondergrens, de entropie $H(X)$. Een lagere entropie betekent dat er meer symbolen gegenereerd worden dan nodig zijn. Men zou ook kunnen zeggen dat de variabelen minder informatie per symbool geven dan mogelijk is. Een variabele X met maximale entropie $H_{max}(X)$ kan $2^{H_{max}(X)}$ symbolen genereren, terwijl er maar $2^{H(X)}$ noodzakelijk zijn. Het verschil tussen de maximale entropie en de entropie noemen wij de *redundantie*, *absolute redundantie* of *overvloedigheid*:

$$R(X) = H_{max}(X) - H(X) \tag{1.21}$$

Soms werkt men met het begrip *relatieve redundantie R* of met het begrip *code-efficiëntie CE*:

$$R = R(X)/H_{max}(X) = 1 - \frac{H(X)}{H_{max}(X)} = 1 - CE \qquad [\%] \tag{1.22}$$

Als de redundantie in procenten wordt gegeven, bedoelt men meestal de relatieve redundantie of code-efficiëntie (deze laatste term wordt vaak gebruikt bij compressiemethoden). De schrijfwijze $R_{orde}(X)$ is een manier om de redundantie per Markov-orde aan te geven.

Voorbeeld 1.9

De geschreven Nederlandse taal heeft een *nulde-orde Markov entropie* van $H_0(X) = 2,35$ bit als wij alleen de letterfrequenties in achtnemen. De *maximale entropie* van het geschreven Nederlands komt op $H_{max}(X) = \mathrm{ld}(26) = 4,7$ bit. Dit levert een nulde-orde redundantie van $R_0(X) = 4,7 - 2,35 = 2,35$ bit. Bij het 'quizzen' blijkt dat de deelnemers al bij $CE = 20\%$ of van de letters in staat zijn om een Nederlandse zin te raden. Als rekening gehouden wordt met alle hogere-orde afhankelijkheden zal de entropie van het Nederlands in de buurt van $H(X) = 0,2 \cdot \mathrm{ld}(26) = 0,94$ bit liggen. Hieruit blijkt dat $R(X) = 0,8 \cdot 4,7 = 3,76$ bit of $R = 80\%$.

De redundantie verlaagt de gevoeligheid voor fouten. Daarentegen kan elke verschrijving een onbedoeld maar een correct gespeld woord in redundantieloze taal opleveren. In een redundantieloze taal is het ontwerpen van een (meerdimensionale) kruiswoordpuzzel gemakkelijk.

In de informatietheorie houdt men zich vooral bezig met het naar behoefte verlagen of verhogen van redundantie. Een van de toepassingen van het verlagen van redundantie is het comprimeren van bestanden. De *compressieverhouding*, de verhouding tussen het gecomprimeerde- en ongecomprimeerde bestand, is afhankelijk van de hoeveelheid redundantie in het originele bestand. Anderzijds, wordt in een storingsrijke omgeving de redundantie vergroot om fouten te voorkomen. In de volgende hoofdstukken zullen wij deze aspecten van de redundantie nader bekijken.

1.6 Samenvatting

De formele resultaten zijn samen te vatten in de volgende vuistregels:

- De informatie over een uitkomst, de selectieve informatie, neemt af met de waarschijnlijkheid van die gebeurtenis;
- De entropie van een variabele neemt af naarmate de waarschijnlijkheid ongelijker verdeeld is over de uitkomsten. Of anders gezegd, naarmate de uitkomsten uniformer verdeeld zijn, des te hoger is de entropie van de variabele.
- De entropie van een variabele neemt af naarmate de gebeurtenissen meer voorspelbaar worden;
- De entropie van een samengestelling van variabelen neemt af naarmate zij afhankelijker van elkaar zijn. Of met andere woorden, naarmate variabelen onafhankelijker zijn , des te hoger is hun samengestelde entropie;
- Het verschil tussen de maximale entropie en de entropie wordt redundantie genoemd.

1.7 Opgaven

1. Een roulettespel heeft een draaischijf met 38 genummerde vakjes: 18 rode, 18 zwarte en 2 groene vakjes. Op de draaiende schijf wordt een balletje geworpen. Als de schijf tot rust komt, zal het balletje in één van de vakjes blijven liggen. Elk vakje heeft evenveel kans om het balletje te vangen. De zwarte vakjes zijn oneven genummerd van $1,3,5,\ldots,35$, de rode vakjes zijn even genummerd van $2,4,6,\ldots,36$ en de twee groene vakjes hebben de 'nummers' 0 en 00. Zodra de croupier de kleur of het nummer van het winnende vakje genoemd heeft, mag er niet meer ingezet worden.

   (a) Hoe groot is de selectieve informatie over elke kleur van het vakje?

   (b) Hoe groot is de gemiddelde informatie over de kleur van het vakje?

   (c) Hoe groot is de gemiddelde informatie over het nummer van het vakje?

   (d) Hoe groot is de conditionele entropie (equivocatie) van de kleur als het nummer van het vakje al bekend is?

   (e) Hoe groot is de conditionele entropie (equivocatie) van het nummer als de kleur van het vakje al bekend is?

   (f) Hoe groot is de gemeenschappelijke entropie van de kleur en het nummer van het vakje?

   (g) Hoe groot is de onderlinge entropie van de kleur en het nummer van het vakje?

   (h) Maak van $H(N,K),H(K),H(N),H(N|K),H(N;K),H(K|N)$ een schema zoals figuur 1.3. Houd rekening met de getalswaarde van de entropieën.

2. Een eenvoudige taal heeft een *alfabet* van 3 letters: $\{A,B,C\}$. De relatieve frequenties van deze letters zijn $p_A = 0,6$, $p_B = 0,1$ en $p_C = 0,3$. De letters treden onafhankelijk van elkaar op.

   (a) Bereken de entropie $H(X)$.

   (b) Bereken de waarschijnlijkheid van alle mogelijke woorden van twee letters.

   (c) Bereken de entropie van deze woorden $H(X,Y)$. Welke relatie heeft zij met de entropie $H(X)$?

   (d) Hoe groot is de absolute redundantie R van deze taal?

3. Een taal met het alfabet $\{0,1\}$ wordt beschreven met de eerste-orde Markov-rij met de volgende overgangskansen $p(1|0) = 2/3$ en $p(0|1) = 1/2$.

   (a) Teken het toestandsdiagram van de Markov-rij (zie figuur 1.5).

   (b) Hoe groot is de informatie van een overgang van 1 naar 0?

   (c) Hoe groot is entropie van een 0 of een 1?

(d) Bereken de kansen van drieletterwoorden die beginnen met een 0 en eindigen op een 1.

(e) Bereken de kans op een drieletterwoord dat eindigt op een 1 als men weet dat de eerste letter een 0 is.

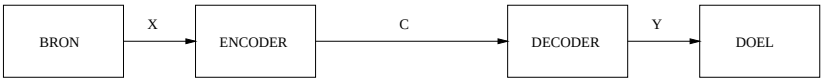
4. Volgens bijlage D komen in normale teksten korte woorden frequenter voor dan lange woorden. In het ‘Groene boekje’ (woordenlijst van de Nederlandse taal) blijkt dat niet te kloppen. Geef hier een verklaring voor.

5. Binary Coded Decimals (‘*BCD*’) is een code waarbij een getal van $0, 1, 2, \ldots, 99$ in een 8 bit-woord (een ‘*byte*’) gecodeerd wordt. Hoeveel redundantie bevat zo’n BCD-woord?

Hoofdstuk 2

Codesystemen voor storingsvrije omgevingen

Een codering is een systeem waarin *bronwoorden* vertaald worden naar *codewoorden* en vice versa. De vertaling van bronwoorden naar codewoorden wordt *codering* of *encodering* genoemd. De terugvertaling van codewoorden naar bronwoorden wordt *decodering* genoemd. De verzameling elementaire uitkomsten - de symbolen of de karakters - wordt het *alfabet* genoemd. Bij bepaalde coderingen zijn het bronalfabet en het codealfabet verschillend.



Figuur 2.1: Het coderingsschema

Bij 100% betrouwbare codering geldt dat $X = Y$. Indien tijdens het terugvertalen van het codewoord naar het bronwoord, het *decoderen*, blijkt dat het terugvertaalde woord Y niet overeenkomt met het originele bronwoord X, noemt men dat een *onbetrouwbare codering*, of een codering met *informatieverlies*. Informatieverlies kan zowel aan interne als aan externe oorzaken liggen. Een externe oorzaak kan bijvoorbeeld de verminking van informatie door een storing zijn (zie hoofdstuk 3). Intern kan informatieverlies zowel bij het encoderen als bij het decoderen optreden. Er zijn ook codesystemen waarin alleen gecodeerd wordt, bijvoorbeeld een boekvertaling. In deze gevallen is het moeilijk de hoeveelheid- en de invloed van het informatieverlies te bepalen:

Voorbeeld 2.1

> Een bepaalde hoeveelheid informatieverlies is acceptabel bij de vertaling van literatuur. Een letterlijke vertaling zou de leesbaarheid niet bevorderen, omdat zinswendingen en woordspelingen vaak taalgebonden zijn.

Soms gebruikt men codering om de *redundantie* te verlagen. Een bronwoord wordt gecodeerd tot een codewoord, dat minder redundantie bevat.

Voorbeeld 2.2

> Bekend is de *morsecode*, waarin de letter 'e' met de hoogste relatieve frequentie, de kortste code krijgt ('.').

Met de verlaging van de redundantie, *compressie*, worden de kosten van opslag en transmissie verlaagd. Afhankelijk van de relatieve frequentie is men in staat codewoorden een zodanige lengte te geven dat de totale lengte van de totale rij gecodeerde woorden minder lang is dan de originele rij woorden.

Compressiemethoden kunnen verdeeld worden in *verliesvrije compressie* - compressie zonder informatieverlies - en *niet-verliesvrije compressie* - compressie met informatieverlies:

Voorbeeld 2.3

> Één fout in de code kan een computerprogramma onbruikbaar maken. Natuurlijk geldt dit ook voor een gecomprimeerd computerprogramma. Hier is alleen verliesvrije compressie mogelijk ('*compaction*').

Voorbeeld 2.4

> Niet alle informatie, verkregen bij een professionele audio- of video-opname, draagt bij aan de verhoging van de afspeelkwaliteit. Om de opslag- en transmissiekosten te drukken, wordt informatieverlies geaccepteerd. Bij dit soort toepassingen wordt vaak gebruik gemaakt van compressie met een acceptabele vervorming die resulteert in verlies aan informatie ('*lossy-compression*').

Een codering kan ook gebruikt worden om de overvloedigheid of *redundantie* te vergroten. Dit is mogelijk door meer codesymbolen gebruiken dan voor de informatie noodzakelijk is. Bij een gegeven entropie $H(X)$ maakt men de maximale entropie $H_{max}(X)$ bewust hoger. Als de informatie wordt verminkt, maakt de redundantie het mogelijk om de verminkte informatie te reconstrueren.

Voorbeeld 2.5

> In celkernen zijn moleculen *DNA* (*deoxyribonucleic acid*) te vinden. Zij bestaat uit zeer 23 chromosoomparen, lange ketens van basenparen. Elk basenpaar wordt aangegeven een van de volgende letters 4: A ('adenine'), T ('thymine'), C ('cytosine') en G ('guanine'). Gemiddeld zijn er 10^4 basenparen in een bacterie en 10^9 in een zoogdier. Een DNA-woord, een '*codon*', bestaat uit 3 letters (basenparen) en codeert een bepaald aminozuur. Er zijn $4^3 = 64$ codes mogelijk, hoewel maar 20 codons voor de aminozuren, 1 startcodon en 1 stopcodon nodig zijn. De overvloedigheid van 42 codons kan gebruikt worden om fouten bij het uitlezen van het DNA te voorkomen. Bijvoorbeeld de code GTC en GTT coderen hetzelfde aminozuur 'glutamine'.

In hoofdstuk 3 zullen codesystemen behandeld worden voor storingsrijke omgevingen. Daarbij speelt de redundantie een belangrijke rol bij het detecteren en herstellen van verminkingen van de informatie.

Tenslotte worden codesystemen ook gebruikt om de informatie te beveiligen tegen onbevoegd gebruik. In hoofdstuk 4 zullen wij hier nader op ingaan.

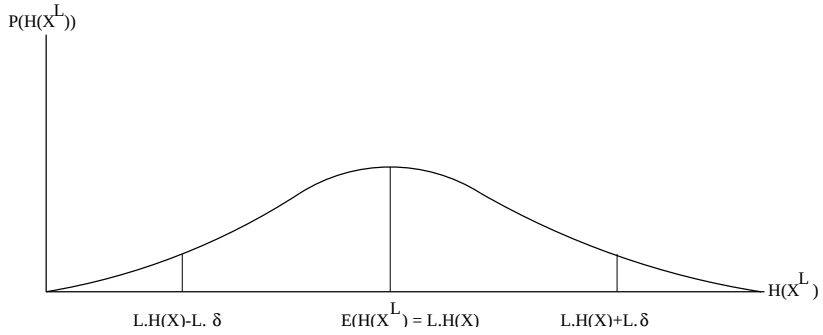
In dit hoofdstuk zullen wij in eerste instantie coderingen voor storingsvrije omgevingen behandelen. Om inzicht in de werking van een codering te krijgen, wordt eerst het *coderingstheorema van Shannon* behandeld voor systemen met een vaste lengte van de bron- en codewoorden. Vervolgens behandelen wij prefixcodes met een vaste lengte van de bronwoorden en een variabele lengte van de codewoorden. Tenslotte worden enkele (merendeels verliesvrije) compressiemethoden behandeld.

2.1 Het broncoderingstheorema van Shannon

> **Broncoderingstheorema:** *"Indien een rij van L onafhankelijke woorden – gegenereerd door een bron met een entropie $H(X)$ – groot genoeg is, kan zij met een van te voren vastgestelde mate van verlies van informatie, gecodeerd worden tot een omvang van $L \cdot H(X)$ bit. Anderzijds, als deze rij van L bronwoorden verder dan $L \cdot H(X)$ bit wordt gecodeerd dan is er sprake van aanzienlijk informatieverlies."* (C. Shannon)

Uit dit theorema blijkt dat het mogelijk is een codering met een vastgestelde mate van verlies, dus ook zonder informatieverlies, te specificeren, mits de lengte van de rij bronwoorden L maar groot genoeg is. Om dit theorema te verklaren, beschouwen wij een bron met een entropie $H(X)$ die L onafhankelijke woorden genereert. Een rij van L woorden en heeft een actuele entropie van $H(X^L)$ bit. De verzameling van alle rijen bronwoorden heeft een gemiddelde entropie $E(H(X^L)) = L \cdot H(X)$ bit (zie formule 1.6). Een bron heeft vaak een niet-uniforme verdeling van woorden zoals bijvoorbeeld figuur 2.2. Bij het coderen selecteert men ongeveer $2^{L \cdot (H(X) \pm \delta)}$ bronwoorden uit de totale verzameling $2^{L \cdot H_{max}(X)}$ bronwoorden.

Sommige bronwoorden worden niet gecodeerd. Hierdoor onstaat de kans op *informatieverlies*, de kans dat een bronwoord niet tot selectie van $2^{L \cdot (H(X) \pm \delta)}$ bronwoorden behoort. Dit is de kans dat de entropie $H(X^L)$ van een bronwoord niet in het gebied $(E(H(X^L) - L \cdot \delta \leq H(X^L) \leq E(H(X^L) + L \cdot \delta)$ valt. Als bekend is dat de entropieën van de bronwoorden normaal verdeeld zijn, is overschrijdingskans te berekenen met een Normaal-verdeling. Maar bij een onbekende kansverdeling, weliswaar met een bekend gemiddelde, staat ons alleen de *ongelijkheid van Chebyshev* (bijlage B) ter beschikking om de kans uit te rekenen dat de rij bronwoorden niet in het interval $(L \cdot H(X) - L \cdot \delta \leq H(X^L) \leq L \cdot H(X) + L \cdot \delta)$ valt::



Figuur 2.2: Kans op informatieverlies

$$\lim_{L\to\infty} P(|H(X^L) - \overbrace{L\cdot H(X)}^{E(H(X^L))}| > L\cdot\delta) < \frac{\overbrace{Var(X^L)}^{L\cdot Var(X)}}{L^2\cdot\delta^2} = \frac{Var(X)}{L\cdot\delta^2} = 0 \tag{2.1}$$

Uit het feit dat de variantie van de bron $Var(X)$ constant is, volgt dat de kans op informatieverlies tot nul nadert indien L groter wordt. Resumerend kan men stellen dat bij toenemende L de entropie $H(X^L)$ van de codewoorden met een de lengte L, nadert tot L keer de gemiddelde entropie van de bronwoorden $L\cdot H(X)$ zonder kans op informatieverlies. Dit lijkt een veelbelovend resultaat. Echter door de toenemende L neemt ook het aantal bronwoorden toe met 2^L. De hoeveelheid bronwoorden en de daaraan toegewezen codewoorden zal door de beperkte opslagruimte en verwerkingstijd beperkt worden. Deze nadelen worden minder als de lengte van een codewoord varieert en afhankelijk wordt gemaakt van de relatieve frequentie.

2.2 Coderingen met variabele lengte

Coderingen van bronwoorden met een vaste lengte met behulp van codewoorden met een variabele lengte, is mogelijk als frequent optredende codewoorden een kortere lengte krijgen dan minder frequent optredende codewoorden. Als voor alle mogelijke $2^{H_{max}(X)}$ bronwoorden een codewoord bestaat, treedt er geen informatieverlies op. Hoewel de gemiddelde lengte van de rij codewoorden korter is dan de oorspronkelijke rij bronwoorden, is er een kleine kans dat de lengte van de rij codewoorden groter wordt.

Indien voor elk bronwoord X_i – met een kans p_i – een codewoord $c_i \in C$ met de lengte $L(c_i)$ bestaat, is een *optimale code* mogelijk als:

$$L(c_i) = -\mathrm{ld}\,(p_i)$$

Bij deze lengte kan de gemiddelde lengte van de codewoorden $E(L(C))$ gelijk worden (met een binair alfabet) aan de entropie van de bron:

$$E(L(C)) \quad = \quad \sum_{i=1}^{|C|} p_i \cdot L(c_i) \quad = \quad \sum_{i=1}^{2^{H_{max}(X)}} -p_i \cdot \mathrm{ld}\,(p_i) \quad = \quad H(X)$$

Dit betekent dat bij een optimale code alle codewoorden uniform verdeeld zijn in de rij codewoorden. Bij de meeste praktische uitvoeringen van optimale codes zal de lengte van de codewoorden een natuurlijk getal zijn met een minimale waarde van 1 en daardoor zal de echte entropie iets hoger zijn dan de optimale waarde $H(X)$. In de praktijk wordt een '*optimale code*' gebonden aan de volgende grenzen:

$$H(X) \leq E(L(C)) < H(X) + 1$$

Indien het code-alfabet $2 < n$ letters heeft, dan wordt de formule voor de begrenzing van de *n-aire entropie* $H(X)/\mathrm{ld}\,(n)$ van de codewoorden:

$$\frac{H(X)}{\mathrm{ld}\,(n)} \leq E(L(C)) < \frac{H(X)}{\mathrm{ld}\,(n)} + 1 \tag{2.2}$$

In de volgende paragraaf gaan wij enkele eigenschappen van zo'n 'optimale code' bestuderen.

2.3 Prefix coderingen

Stel dat voor een alfabet met 4 letters $[A,B,C,D]$ met de relatieve frequentie $[\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}]$ gecodeerd moeten worden met het binaire alfabet $[0,1]$. Wij zouden deze code op verschillende manieren kunnen maken. De verschillende codes vindt u in de kolommen die genummerd zijn van 1 tot 4:

bronwoord			codewoord			
i	inhoud	p_i	1	2	3	4
1	A	0,5	00	0	0	0
2	B	0,25	01	01	01	10
3	C	0,125	10	10	011	110
4	D	0,125	11	11	111	111
$E(C(L))$		$\sum p_i \cdot L(c_i)$	2,0	1,5	1,75	1,75
"DACAB" gecodeerd:			1100100001	11010001	1110011001	1110110010
goed gedecodeerd:			"DACAB"	"DACAB"	"DACAB"	"DACAB"
fout gedecodeerd:				"DBAAB"		

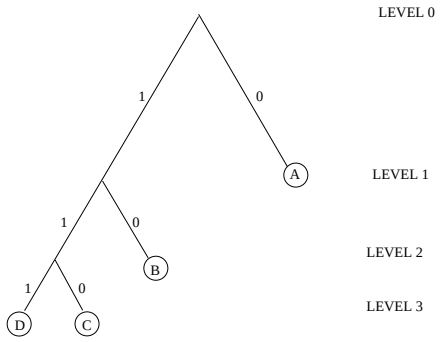
1. Code 1 heeft geen leestekens nodig om de codewoorden van elkaar te onderscheiden omdat elk codewoord met 2 computerbits gecodeerd is;

2. Hoewel code 2 het meest efficiënt lijkt, bestaat de mogelijkheid dat een bericht op niet-unieke wijze gedecodeerd wordt. Daarom zijn bij deze code leestekens noodzakelijk en verdwijnt het voordeel van de efficiënte codering;

3. Code 3 heeft geen leestekens nodig omdat zij uniek decodeerbaar is. Toch heeft deze code een verborgen nadeel. De deelrij 110011... is dubbelzinnig omdat zij "DAC" of "DAAD" kan opleveren. Deze dubbelzinnigheid verdwijnt pas bij het inlezen van het volgende bit. Dit is niet gunstig omdat een dubbelzinnige deelrij tijdelijk opgeslagen moet worden;

4. De beste code is code 4. Dit is een code die geen leestekens nodig heeft omdat zij uniek decodeerbaar is. Bovendien is zij direct decodeerbaar zonder tijdelijke opslag van dubbelzinnige toestanden. Deze code is een *prefixcode*.

Prefixcodes zijn het meest geschikt voor discrete codering omdat zij uniek en direct decodeerbaar zijn en toch efficiënt zijn. Wij kunnen ons afvragen of het altijd mogelijk is om k bronwoorden prefix te coderen met een gelijk aantal codewoorden, als de codewoorden c_i een lengte $L(c_i) \geq -\mathrm{ld}\,(p_i) < L(c_i) + 1$ hebben. Om te analyseren of zo'n prefixcode mogelijk is, zullen wij code 4 weergeven als een *prefixboom* (figuur 2.3).



Figuur 2.3: De boom van de binaire prefix code 4.

Intuïtief is duidelijk dat zo'n prefixcode gestructureerd is als een boom met keuzes. De binaire prefixboom van figuur 2.3 heeft 4 levels: $0 \ldots 3$. Elk *level* kan een aantal knopen en bladeren bevatten. Elke knoop staat voor een *keuze*, elk blad staat voor een codewoord c_i. Een knoop kan dus nooit voor een codewoord staan. De lengte van het pad van de *wortel* naar het betreffende blad is gelijk aan het level en de lengte L_i van het codewoord.

Het maximaal aantal bladeren in een binaire prefixboom is gelijk aan $2^{L_{max}}$. Indien een blad op een tussen gelegen level $L(c_i)$ geplaatst wordt, dan wordt de prefixboom gesnoeid en raakt $2^{L_{max}-L(c_i)}$ bladeren kwijt. Het aantal gesnoeide k bladeren is maximaal $2^{L_{max}}$ (codewoorden):

$$\sum_{i=1}^{k} 2^{L_{max}-L(c_i)} \leq 2^{L_{max}}$$

Hieruit volgt de belangrijkste voorwaarde voor het bestaan van een *binaire prefixboom* indien het aantal codewoorden $|C|$ met hun lengte $L(c_i)$ gegeven is:

$$\sum_{i=1}^{|C|} 2^{-L(c_i)} \leq 1$$

Voor een *n-air code-alfabet* wordt de algemene formule voor een prefixboom:

$$\sum_{i=1}^{k} n^{-L(c_i)} \leq 1 \tag{2.3}$$

Als een code niet aan formule 2.3 voldoet, is zij geen prefixcode. Het is niet zo dat een code die wel aan deze voorwaarde voldoet een prefixcode is. Voor de code 2 geldt dat $2^{-1}+2^{-2}+2^{-2}+2^{-2} = 1,25$ niet voldoet. Code 3 en 4 voldoen wel: $2^{-l_i} = 2^{-1}+2^{-2}+2^{-3}+2^{-3} = 1$. Toch is code 3 geen binaire prefixcode.

Wanneer is er sprake van een optimale prefixcode? Bij een optimale prefixcode zal de lengte $L(c_i)$ van een codewoord c_i zoveel mogelijk gelijk moeten zijn met de waarde $-\mathrm{ld}(p_i)$. Men kan een optimale prefixboom beschouwen als een soort 'mobiel', bekend uit de kinderkamer. De gewichten stellen de codewoorden voor. De gehele mobiel moet zoveel mogelijk in evenwicht zijn. Uit de het coderingstheorema van Shannon kan men afleiden dat bij verliesvrije codering de gemiddelde entropie van de codewoorden $E(L(C))$ in ieder geval groter of gelijk moet zijn aan de bronentropie $H(X)$. Als wij een maximale afwijking van 1 bit accepteren, dan geldt bij een *binair alfabet* voor de gemiddelde lengte van de codewoorden $E(L(C))$:

$$H(X) \leq E(L(C)) < H(X)+1 \tag{2.4}$$

Als er gebruik wordt gemaakt van een *n-air code-alfabet*, dan verandert deze formule in:

$$\frac{H(X)}{\mathrm{ld}(n)} \leq E(L(C)) < \frac{H(X)}{\mathrm{ld}(n)}+1 \tag{2.5}$$

2.4 De Shannon-Fanocode

Bij de *Shannon-Fanocode* wordt een codewoord c_i met de relatieve frequentie p_i gehangen aan een blad in de prefixboom op level $-\lceil \mathrm{ld}(p_i) \rceil$ (de kleinste integer waarde die even groot of groter is dan de waarde $-\mathrm{ld}(p_i)$). Omdat $-\mathrm{ld}(p_i) \leq -\lceil \mathrm{ld}(p_i) \rceil < -\mathrm{ld}(p_i)+1$ geldt:

$$H(X) = -\sum_{i=1}^{k} p_i \cdot \mathrm{ld}\,(p_i) \leq \overbrace{-\sum_{i=1}^{|C|} p_i \cdot \lceil \mathrm{ld}\,(p_i) \rceil}^{E(L(C))} < -\sum_{i=1}^{k} p_i \cdot (\mathrm{ld}\,(p_i)+1) = H(X)+1$$

Voorbeeld 2.6

De Shannon-Fanocode van 5 bronwoorden met de relatieve frequenties

$$[0,15 \;\; 0,25 \;\; 0,32 \;\; 0,12 \;\; 0,16]$$

geeft de volgende codewoordlengten:

$$\begin{array}{lll}
-\mathrm{ld}\,(0,15) = 2,7 \leq L(c_1) < -\mathrm{ld}\,(0,15)+1 = 3,7 & \rightarrow & L(c_1) = 3 \\
-\mathrm{ld}\,(0,25) = 2,0 \leq L(c_2) < -\mathrm{ld}\,(0,25)+1 = 3,0 & \rightarrow & L(c_2) = 2 \\
-\mathrm{ld}\,(0,32) = 1,6 \leq L(c_3) < -\mathrm{ld}\,(0,32)+1 = 2,6 & \rightarrow & L(c_3) = 2 \\
-\mathrm{ld}\,(0,12) = 2,6 \leq L(c_4) < -\mathrm{ld}\,(0,12)+1 = 3,6 & \rightarrow & L(c_4) = 3 \\
-\mathrm{ld}\,(0,16) = 2,6 \leq L(c_5) < -\mathrm{ld}\,(0,16)+1 = 3,6 & \rightarrow & L(c_5) = 3
\end{array}$$

Voor deze codewoorden is een prefixcode mogelijk: $2^{-3}+2^{-2}+2^{-2}+2^{-3}+2^{-3} = 7/8$ De overeenkomstige 5 codewoorden zijn: [110 01 00 101 100]. De gemiddelde lengte van de codewoorden is $E(L(C)) = \sum p_i \cdot L(c_i) = 2,43$ bit, de bronentropie is $H(X) = -\sum_{i=1\ldots k} p_i \cdot \mathrm{ld}\,(p_i) = 2,23$ bit. De redundantie komt op $R(X) = E(L(C)) - H(X) = 2,43 - 2,23 = 0,20$ bit.

De Shannon-Fanocode heeft een belangrijk bezwaar: Aan de ergste '*uitschieter*', dit is het codewoord met de laagste kans, wordt vaak een grotere lengte toegekend dan noodzakelijk is. Om aan dit bezwaar tegemoet te komen, is de *Huffmancode* ontwikkeld. In de Huffmancode wordt een uitschieter zoveel mogelijk in de buurt van het op-een-na langste codewoord geplaatst. De Huffmancode komt daarmee nog dichter bij de optimale entropie.

2.5 De Huffmancode

Het verkorten van de uitschieters met het algoritme van Huffman gaat door codewoorden zo dicht mogelijk bij de wortel en zoveel mogelijk in het zelfde level te plaatsen. Daarom worden eenheden met een lage waarschijnlijkheid gecombineerd tot eenheden met een hogere waarschijnlijkheid.

> *"Neem de twee bronwoorden met de laagste relatieve frequenties. Deze twee bronwoorden krijgen de langste codewoorden, even groot maar verschillend in het laatste codesymbool. Vervang deze twee bronwoorden door een nieuw bronwoord met de som van de frequenties en herhaal de procedure."*

Indien men dit bottom-up algoritme zou moeten programmeren, zou een *priority queue* een zeer geschikte datastructuur zijn. Plaats eerst alle bronsymbolen in de priority queue. Verwijder de twee bronsymbolen met de laagste prioriteit, de laagste waarschijnlijkheid, uit de priority queue. Combineer de twee codewoorden tot een eenheid waarvan de gecombineerde waarschijnlijkheid de som is van de samenstellende codewoorden. Plaats de combinatie terug in de priority queue en herhaal de procedure.

Het bepalen van een Huffmancode van bronwoorden met een lengte van L bit heeft een tijd- en ruimtecomplexiteit van $O(2^L \log 2^L)$. In de praktijk beperkt men zich daarom tot bronwoorden met een lengte van niet meer dan 24 bit.

Voorbeeld 2.7

De Huffmancode van 5 bronwoorden met de relatieve frequenties $[0,15\ \ 0,25\ \ 0,32\ \ 0,12\ \ 0,16]$ geeft de volgende samenvoegingen $[[[0,12+0,15]+0,32]+[0,16+0,25]]$ waaruit de volgende codewoorden ontstaan: $[101\ \ 10\ \ 11\ \ 100\ \ 00]$ Voor deze codewoorden is een prefixcode mogelijk: $2^{-3}+2^{-2}+2^{-2}+2^{-3}+2^{-2}=1$. De redundantie wordt $R(X)=E(L(C))-H(X)=2,27-2,23=0,05$ bit.

Voorbeeld 2.8

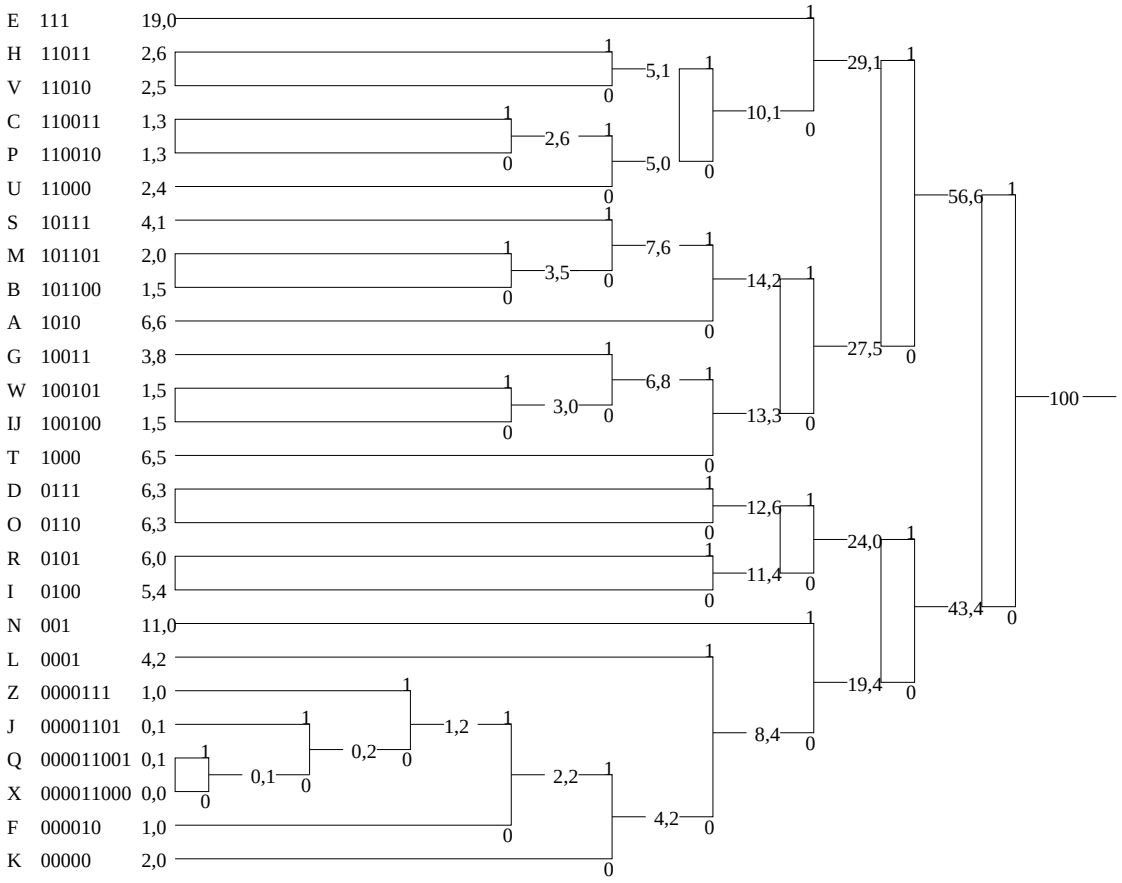
Een binaire Huffmancode is gebaseerd op bronwoorden van één bronsymbool (één letter) en de letterverdeling van de Nederlandse taal (zie bijlage C). Het algoritme van Huffman leidt tot de volgende boom:

Deze Huffmancode met Nederlandse letterverdeling komt uit op een nulde-orde redundantie $R_0(X)$ van $4,700-4,115=0,585$ bit. In de praktijk is de besparing groter omdat een bronsymbool 7 of 8 bit in *ASCII-code* kost. In het laatste geval wordt een *compressieverhouding* van $0,4115/8=0,51$ gerealiseerd (exclusief de frequentie informatie).

Niet alleen uit theoretische overwegingen maar ook uit de twee voorbeelden in bijlage C, blijkt dat de Huffmancode een betere benadering geeft van de optimale entropie dan de Shannon-Fanocode. De gemiddelde lengte van codewoorden wordt bij de Huffmancode $4,115$ bit en bij de Shannon-Fanocode $4,124$ bit.

2.6 Compressiemethoden

Compressie wordt toegepast bij opslag en verzending van rijen bronwoorden. Theoretisch stelt het coderingstheorema van Shannon dat de redundantie, zonder verlies aan



Figuur 2.4: Huffmanboom voor de Nederlandse taal

informatie verlaagd kan worden. Gaat men verder dan de maximale redundantie, dan is informatieverlies onvermijdelijk.

Hoewel het coderingstheorema van Shannon langere bronwoorden adviseert, zal de lengte van bronwoorden (een of meer bytes, cijfers of letters) bij beperkte opslag- en rekentijd niet te groot kunnen worden. Het is haast paradoxaal dat bij een hogere opslagcapaciteit nog meer reductie mogelijk is. Hoewel er nog veel hogere-orde redundantie achterblijft, wordt bij het comprimeren van natuurlijke talen de lengte van de bronwoorden niet alleen om capaciteitsredenen beperkt tot 3 à 4 letters. De kans op een spatie of een leesteken neemt boven deze lengte sterk toe (zie bijlage D).

De meeste compressiemethoden trachten binnen beperkte tijd tot redelijke resultaten te komen, zal bij onbekende frequentieverdelingen van de bronwoorden zal het codeerproces eerst de frequenties van de bronwoorden bepalen. Nadat deze frequentieverdeling bekend is, worden de bronwoorden vervangen door codewoorden. Het vooraf volledig bepalen van de frequentieverdeling kan het codeerproces bij grote rijen bronwoorden inefficiënt maken en blijkt niet echt noodzakelijk te zijn.

Omdat de frequentieverdeling ook bij het decoderen bekend moet zijn, moet zij samen met de gecomprimeerde rij worden aangeleverd. Vooral bij kleine rijen resulteert dit in een beduidend lagere *compressieverhouding*.

Deze uitvoeringsproblemen hebben geleid tot de ontwikkeling van *compressiemethoden* die dynamisch in één fase de frequentieverdeling en de vertaling van de rij bronwoorden bepaald. Bij de *adaptieve compressiemethoden* wordt in eerste instantie voor alle bronwoorden een uniforme frequentieverdeling aangenomen. Tijdens het encodeer- en het decodeerproces worden dezelfde relatieve frequenties berekend. Omdat het decodeerproces van dezelfde frequentieverdeling uitgaat, komt zij tot dezelfde frequentieverdeling van de bronwoorden als het encodeerproces. Voorbeelden van adaptieve hogere-orde compressiemethoden zijn '*pkzip*', '*gzip*' (het Lempel-Ziv algoritme). Bijzonder efficiënte adaptieve hogere-orde compressiemethoden zijn *bzip2* en *arithmetic codering*.

Naast adaptieve en niet-adaptieve compressiemethoden bestaat een belangrijk onderscheid tussen verliesvrije en niet-verliesvrije compressiemethoden.

2.7 Verliesvrije compressie

Bij *verliesvrije compressie* of *compactie* is de gedecomprimeerde rij symbolen gelijk aan de originele rij symbolen en moet de verhouding tussen de gecomprimeerde rij en de originele rij, de *compressieverhouding* kleiner zijn dan 100%. Bovendien moet elke gecomprimeerde rij uniek zijn (het *uniciteitsprincipe*) omdat er anders dubbelzinnigheid ontstaat bij het decomprimeren. Deze bovenstaande eisen lijken voldoende maar bij nadere overweging moeten wij ook rekening houden met de omvang van het decompressieprogramma:

Voorbeeld 2.9

> Stel dat de inhoud van "Het ontstaan van soorten" van Charles Darwin wordt gecomprimeerd tot een '1', De "Bijbel" wordt gecomprimeerd tot een '2' en de "Inleiding in de Informatietheorie" tot een '3'. Hoewel de gecomprimeerde code zeer kort is, moet het decompressie-programma de beschikking hebben over de volledige inhoud van deze drie werken.

Voorbeeld 2.10

> Bij het decomprimeren van een Huffmancodering moeten wij naast de gecomprimeerde rij in principe ook de beschikking hebben over de Huffmanboom.

Voorbeeld 2.11

> Bekend zijn de *zelfuitpakkende bestanden*. Deze bestanden bevatten het gecomprimeerde bestand en het decompressieprogramma.

Zijn alle rijen zonder verlies te comprimeren? Nee, het aantal rijen minder dan n bit is kleiner dan het aantal rijen van exact n-bit:

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1 \leq 2^n$$

Het '*duiventilprincipe*' stelt dat één van de 2^n rijen van n bit niet uniek te comprimeren is. Deze beperking geldt ook voor niet-binaire symbolen. Uit deze redenering blijkt wel dat dit theoretisch bezwaar bij '*lossy compression*', door het afzien van het uniciteitsprincipe, niet meer geldt.

Voorbeeld 2.12

> Een gecomprimeerd bestand kan zelfs groter worden als wij het nogmaals trachten te comprimeren.

Tot hoever is compressie zonder verlies nog mogelijk? De bewering dat het mogelijk is een volledig boek zonder verlies aan informatie in een paar bit op te slaan, is onwaarschijnlijk. Een beter antwoord op deze vraag komt uit de *algoritmische informatietheorie* van Andrej Kolmogorov. Volgens deze theorie wordt de ondergrens van een compressie bepaald door de lengte van het kleinst mogelijke programma dat de ongecomprimeerde rij kan reconstrueren. De lengte van dit theoretische reconstructieprogramma $K(x)$ (in bit) wordt '*Kolmogorov Complexiteit*' genoemd.

Voorbeeld 2.13

> Een rij die uit 10^6 nullen bestaat zoals $000\ldots000$ kan worden uitgevoerd door het volgende korte reconstructieprogrammaatje:
>
> **for** $i = 1,\ 1000000$ **print** '0'

Voorbeeld 2.14

> Een rij zonder herkenbaar patroon kan alleen door expliciet uitschrijven worden gegeven, het reconstructieprogramma is relatief groter dan de vorige voorbeelden:
>
> **print** '10101011011100011111011101100000101101010'

Uit deze voorbeelden blijkt dat hoe willekeuriger het patroon, des te groter het reconstructieprogramma. Door nieuwe patronen te zoeken zal men kortere reconstructieprogramma's trachten te maken, omdat het kleinst mogelijke reconstructieprogramma meestal niet bekend is. Volgens de algoritmische informatietheorie is een rij x niet meer comprimeerbaar als haar lengte plus de lengte van het decomprimeerprogramma kleiner wordt dan de lengte het van het kleinst mogelijke reconstructieprogramma $K(x)$:

$$K(x) > lengte(x) + C \tag{2.6}$$

De waarde C staat voor de invloed van de computertaal en de apparatuur op het decomprimeerprogramma die onafhankelijk van de lengte van de rij is. Uiteindelijk blijkt bij toenemende lengte van rij en reconstructieprogramma, de invloed van de constante C in formule 2.6 te verdwijnen.

Zijn niet-comprimeerbare rijen berekenbaar? Met andere woorden, is er een programma dat niet-comprimeerbare rijen kan vinden? Of is er een intelligent programma dat van een willekeurige rij een decompressieprogramma met een lagere complexiteit kan vinden? Neem aan dat het wel zo is, dan is er een eindig programma P dat een niet-comprimeerbare rij Q berekent uit alle rijen die mogelijk zijn bij n bit:

$$\textbf{programma}(n)\{...\}$$

Bij groter wordende rijen $n \to \infty$ kan dit 'eindige' programma p uiteindelijk kleiner gecomprimeerd worden dan een niet-comprimeerbare rij. Dit leidt tot tegenspraak, waaruit blijkt dat zo'n programma niet kan bestaan. Alleen niet-algoritmische methoden zullen in staat zijn om betere compressiemethoden te vinden. Bovendien kunnen wij concluderen dat bij elk compressie-programma niet-comprimeerbare rijen zijn!

2.7.1 'Run-Length' codering

De effectiviteit van deze compressiemethode is zeer afhankelijk van de aard van de bronrij. *Run-Length codering* wordt meestal gebruikt voor de codering van beelden waarbij snelle verwerking noodzakelijk is en bij ijle ('*sparse*') bitarrays. Het algoritme is zeer simpel qua ontwerp en efficiënt qua uitvoering:

Vervang een deelrij van 10 nullen door 1010 (het getal 10 in binaire code). Een rij met afwisselend 'nullen' en 'enen':

$$11100000000001111111100001100000011111111111$$

dit wordt met Run-Length codering:

$$11, 1010, 1000, 100, 10, 101, 1011$$

Vaak wordt een Run-Length codering weer gecodeerd met een andere compressiemethode zoals een nulde-orde Huffman codering. Run-Length codering wordt veel toegepast bij de *FAX*.

2.7.2 'Move-to-Front' codering

Een methode die enigszins op de Run-Length codering lijkt, is de '*Move-to-Front*' codering. Deze codering is geschikt voor bronrijen waar veel gelijke symbolen dicht bij elkaar staan, zoals woordenboeken en gesorteerde bestanden. De methode maakt gebruik van een rij van m alfabetsymbolen. Bij aanvang van de compressie zijn alle alfabetsymbolen in volgorde in de rij A opgenomen. De volgorde van de alfabetsymbolen is genummerd van $1, \ldots m$. Zodra het eerste symbool van de bronrij B wordt gelezen, wordt zijn relatieve plaats in de alfabetrij A het codesymbool. Bovendien wordt het actuele alfabetsymbool vooraan in de alfabetrij A geplaatst en verkrijgt het voortaan een kortere code (i.v.m. een hogere frequentie). Dit proces herhaalt zich tot en met het laatste symbool van de bronrij B. Het zal duidelijk zijn dat als het alfabet groot genoeg is en veel gelijke symbolen dicht bij elkaar staan de Move-to-Front codering uit relatief veel kleine getallen bestaat. Dit maakt het zeer geschikt als een voorbereidende fase op een Huffman codering.

Voorbeeld 2.15

Een taal met $m = 3$ alfabetsymbolen 'b', 'o' en 'r' heeft de bronrij '*oorrob*' gegenereerd. De compressie start met $A = [b, o, r]$

broncode	A	mfcode	hmcode
o	[b,o,r]	2	10
o	[o,b,r]	1	0
r	[o,b,r]	3	110
r	[r,o,b]	1	0
o	[r,o,b]	2	10
b	[o,r,b]	3	110

Tabel 2.1: 'Move-to-Front' (mfcode) met Huffman (hmcode).

De 'Move-to-Front' codering heeft van de bronrij '*oorrob*' de coderij '213123' gemaakt (Huffman '100110010110')

Het decomprimeren van een Move-to-Front codering geschiedt op vergelijkbare wijze:

Voorbeeld 2.16

Een taal met $m = 3$ alfabetsymbolen 'b', 'o' en 'r' heeft met 'Move-to-Front' codering de rij '213123' gevormd en deze met Huffman tot de rij '100110010110' gecodeerd. Na het decoderen van de Huffmancode start de decompressie met de rij alfabetsymbolen $A = [b, o, r]$:

De decodering heeft de rij '$10, 0, 110, 0, 10, 110$' met Huffman gedecodeerd tot de rij '213123'. Met Move-to-Front decodering is uit deze laatste rij de bronrij '*oorrob*' gevonden.

hmcode	mfcode	A	broncode
10	2	[b,o,r]	o
0	1	[o,b,r]	o
110	3	[o,b,r]	r
0	1	[r,o,b]	r
10	2	[r,o,b]	o
110	3	[o,r,b]	b

Tabel 2.2: 'Move-to-Front' decodering.

2.7.3 Het Lempel-Ziv algoritme

In de computertechniek wordt regelmatig gebruik gemaakt van het 'one-pass' *Lempel-Ziv algoritme* voor verliesvrije compressie. Dit algoritme vervangt deelrijen door verwijzingen naar eerder opgetreden overeenkomstige deelrijen. Hoewel de compressieprestatie beperkt is, kan het zeer efficiënt geïmplementeerd worden (in $O(L)$ voor tijd- en geheugengebruik bij bronwoorden van L bit) indien de deelrijen opgeslagen en gevonden worden in $O(1)$ tijd, bijvoorbeeld door gebruik te maken van een hash-tabel (dictionary) voor de verzameling deelrijen.

Het 'one-pass' Lempel-Ziv algoritme gaat als volgt:

1. Initialiseer de verzameling deelrijen D met alle deelrijen met de lengte 1 (meestal het alfabet) $D = \{0, 1\}$, geef elk toegevoegd element een oplopende index;
2. 'Match' de langst mogelijke deelrij in de verzameling deelrijen aan de kop van de onverwerkte invoerrij;
3. Plaats de index van de deelrij in de uitvoerrij;
4. Plak rechts aan de deelrij het volgende karakter uit de invoerrij vast, plaats deze nieuw gevormde deelrij in de verzameling deelrijen met de hoogste $index + 1$;
5. Zolang er nog onverwerkte invoerkarakters zijn moet het algoritme vanaf instructie 2 herhaald worden.

De verzameling deelrijen kan na compressie verwijderd worden, omdat het decompressie-algoritme op dezelfde wijze uit de gecomprimeerde rij een identieke verzameling deelrijen opbouwt.

invoerrij:	a	b	b	a	ab	ba	ab	abb	aa	aa	baa	bb	a
uitvoerrij:	0	1	1	0	2	4	2	6	5	5	7	3	0

De letters *a* en *b* vormen de invoerrij, de indexen $0 \ldots 7$ vormen de uitvoerrij. Tijdens het algoritme wordt de verzameling deelrijen als volgt opgebouwd. Niet elke deelrij wordt gebruikt bij de compressie:

rotaties					
o	o	r	r	o	b
b	o	o	r	r	o
o	b	o	o	r	r
r	o	b	o	o	r
r	r	o	b	o	o
o	r	r	o	b	o

Tabel 2.3: Rotaties van de rij '*oorrob*'.

index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
deelrij:	a	b	ab	bb	ba	aa	abb	baa	aba	abba	aaa	aab	baab	bba

Het Lempel-Ziv algoritme (*LZ* of *LZ77*) uit 1977 is in 1978 verbeterd door *Welch*. Het staat nu bekend onder de naam *Lempel-Ziv-Welch algoritme* (*LZW* of *LZ78*). Het patentrecht van LZW is in bezit van Unisys.

Het bekende '*gzip*' (GNU zip) programma is gebaseerd op het (vrij te gebruiken) LZ-algoritme. Het programma '*gzip*' gebruikt blokverwijzingen (een samenstelling van een pointer- en een lengte) om te verwijzen naar een vorige deelrij. In plaats van een deelrij wordt de blokverwijzing in de uitvoer opgenomen. De verwijzing beperkt zich tot afstanden minder dan 32 Kbytes, de bloklengte beperkt zich tot 256 bytes. Wanneer voor een deelrij geen verwijzing gemaakt of gevonden kan worden, dan wordt zij ongewijzigd in de uitvoer opgenomen.

2.7.4 Compressie met bloksortering

Bloksortering is gebaseerd op het *Burrows-Wheeler* '*bloksorterings-algoritme*'. Dit algoritme permuteert een rij symbolen, waardoor hogere orde redundantie gemakkelijker wordt verkregen. Het bekende compressieprogramma '*bzip2*' is gebaseerd op het bloksorterings-algoritme gevolgd door een Huffmancodering. Met '*bzip2*' kan men een zeer lage compressieverhouding krijgen in een korte verwerkingstijd. Zover bekend, worden door het programma '*bzip2*' geen patentrechten geschaad en mag het vrij gebruikt worden. Wij zullen het bloksorterings-algoritme demonstreren aan de hand van het volgende voorbeeld:

Voorbeeld 2.17

Het bloksorterings-algoritme begint met het maken van een verzameling van cyclische permutaties (rotaties) van de bronsymbolen. De rij '*oorrob*' van 6 symbolen levert de volgende tabel met rotaties op:

De 6 rotaties worden lexicografisch gesorteerd. Het resultaat van deze lexicografische bloksortering is een tabel, waarbij de laatste kolom, opgebouwd uit de laatste symbolen van de rotaties '*orboro*', een permutatie is van '*oorrob*'. De rij '*oorrob*'

rij	rotaties					
1	b	o	o	r	r	o
2	o	b	o	o	r	r
→3	o	o	r	r	o	b
4	o	r	r	o	b	o
5	r	o	b	o	o	r
6	r	r	o	b	o	o
laatste kolom ↑						

Tabel 2.4: Gesorteerde rotaties.

blijkt in deze tabel op rij 3 te staan. De index van deze rij moeten wij bewaren omdat zij nodig is bij het decomprimeren.

Het resultaat van de bloktransformatie van '*oorrob*' is de kolom '*orboro*' en de rij-index 3.

Wanneer bijvoorbeeld een Nederlandse tekst op deze manier gepermuteerd wordt, zullen bijvoorbeeld alle rotaties die beginnen met een 'ij' in de lexicografische sortering vooraan naast elkaar geplaatst staan. De letters die frequent vooraf gaan aan de letter-combinatie 'ij' zoals de *h*, *m* en een *z* ('hij', 'mij' en 'zij') zullen in de laatste kolom, de resulterende permutatie, dicht bij elkaar staan. Dit effect treedt op voor alle hogere--orde afhankelijkheden tussen de bronsymbolen in *B*. Het resultaat is dat tengevolge van hoge-orde redundantie veel dezelfde symbolen in de laatste kolom bij elkaar staan. Deze hoge-orde redundantie kan efficiënt gecodeerd worden met een Move-to-Front codering. Daarom wordt het resultaat van het bloksorteringsalgoritme, de laatste kolom van tabel 2.4, gecomprimeerd met een nulde-orde Huffmancodering nadat zij met een Move-to-Front codering is vertaald.

Het decompressieproces zal de nulde-orde Huffman code weer vertalen in de Move-to-Front codering, waaruit vervolgens de laatste kolom van tabel 2.4 weer teruggevonden wordt. Omdat bij het herstel van de bronrij de laatste kolom en de rij-index noodzakelijk zijn, is de rij-index van de originele bronrij in tabel 2.4 ook in het compressiebestand opgenomen.

Voorbeeld 2.18

Tijdens het terugtransformeren wordt de bronrij '*orboro*' in de laatste kolom van de nieuwe tabel 2.5 geplaatst. Vervolgens wordt de kolom '*orboro*' op letters gesorteerd om de kolom '*booorr*' te vinden. Dit blijkt de eerste kolom van de tabel van gesorteerde rotaties (zie: tabel 2.4). Omdat de kolom '*orboro*' in de cyclische permutaties een positie vooraf gaat aan de kolom '*booorr*' plaatsen wij in de nieuwe tabel de kolom '*booorr*' voor de kolom '*orboro*':

De twee kolommen 1 en 2 zijn permutaties van elkaar. Wij moeten nu een inverse permutatie T vinden die kolom 2 op kolom 1 afbeeldt. Deze inverse permutatie T

	kolom	
rij	1	2
1	o	b
2	r	o
3	b	o
4	o	o
5	r	r
6	o	r

Tabel 2.5: Laatste- (1) en eerste kolom (2).

i	T(i)
1	3
2	1
3	4
4	6
5	2
6	5

Tabel 2.6: Inverse permutatie $T(i)$.

wordt als volgt gevonden:

Wij beginnen bij de eerste rij in kolom 2. De letter '*b*', staat ook op rij 3 van kolom 1. Daaruit volgt dat $T(1) = 3$. Vervolgens blijkt de letter '*o*' op tweede rij van kolom 2 op de 1-ste, 2-de en 3-de rij van kolom 1 te staan. Wij kiezen voor rij 1 in plaats van een lagere rij omdat wij weten dat deze twee kolommen ontstaan zijn uit een lexicografische sortering van de bronrij. De functie $T(2) = 1$. Vervolgens berekenen wij op dezelfde manier $T(3)$ tot en met $T(5)$:

Als wij deze inverse permutatie $T(i)$ gebruiken op de laatste kolom '*orboro*' uit tabel 2.4 dan vinden wij de voorlaatste kolom '*rroboo*' uit deze zelfde tabel. Na *n* keer terugroteren zal de eerste kolom van tabel '*booorr*' weer ontstaan. Uiteindelijk hebben wij tabel 2.4 volledig teruggevonden. De bronrij '*oorrob*' is nu gemakkelijk te vinden op regel 3 van deze tabel.

De compressie met het '*bzip2*' programma is meestal beter dan het '*gzip* programma. De verwerkingssnelheid van '*bzip2*' is iets lager dan het '*gzip*' of het '*pkzip*' programma, maar beter dan de meeste andere compressieprogramma's.

2.7.5 Arithmetische codering

Zeer goede compressieresultaten worden behaald met *arithmetische codering*. Deze compressiemethode is geschikt voor hoger-orde compressie en is bovendien erg efficiënt $O(L)$.

Dit komt vooral omdat de bronentropie $H(X)$ wordt benaderd met reële waarden in plaats van geheeltallige waarden zoals bij Huffman. Het *algoritme van Elias* (patentrechten in bezit van *IBM, AT&T* en *Mitsubishi*) leidt, via successievelijke partitionering van een reëel interval, uiteindelijk tot het comprimeerde resultaat in de vorm van een 'insluiting' van de binaire breuk $q = 0, f$.

Het algoritme verdeelt het reële interval $[0 \ldots 1]$ in m partities, in vaste volgorde gelabeld van $a_1 \ldots a_m$. Elke partitie a_i is één op één gekoppeld aan een alfabetsymbool $a_{i=1\ldots m}$. Bovendien is de lengte van een partitie evenredig met de waarschijnlijkheid van het bijbehorende alfabetsymbool. Omdat bij de start van de compressie elk symbool a_i even waarschijnlijk is, hebben alle partities a_i in eerste instantie gelijke afmetingsverhoudingen: $p_0(a_1) = p_0(a_2) = \ldots = p_0(a_m) = 1/m$. In stap n wordt, als uitkomst x_n gelijk is aan symbool a_i (symbolisch: $x_n = a_i$), de a_i^{de} partitie opnieuw verdeeld in m partities die weer in vaste volgorde gekoppeld worden met $a_1 \ldots a_m$. Deze nieuwe partitieverdeling heeft ook nieuwe afmetingsverhoudingen $p_n(a_1), p_n(a_2), \ldots, p_n(a_m)$. Elke afmetingsverhouding $p_n(a_i)$ in stap n is gebaseerd op het aantal keren $F_n(a_i)$ dat het symbool a_i al is opgetreden in de rij van n uitkomsten $[x_1, x_2, \ldots x_n]$:

$$F_n(a_i) = \begin{cases} 0 & n = 0 \\ F_{n-1}(a_i) & x_n \neq a_i \\ F_{n-1}(a_i) + 1 & x_n = a_i \end{cases}$$

De afmetingsverhouding $p_n(a_i)$ van de partitie a_i tijdens stap n is:

$$p_n(a_i) = \frac{1 + F_n(a_i)}{\sum_{j=1}^{m}(1 + F_n(a_j))}$$

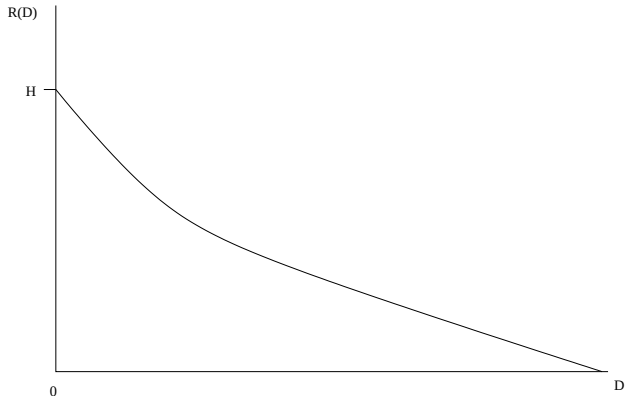
Wij kunnen het interval $[0 \ldots 1]$ verdelen in 2^k gelijke intervallen, genummerd van $f_n = 0 \ldots 2^k - 1$. Als een partitie tijdens stap n volledig valt binnen het interval f_n dan is de bit-rij die binnen f_n valt, het resultaat van de compressie tot en met stap n. Bij elke nieuwe stap $n+1$ kunnen alleen maar laagwaardige bits toegevoegd worden aan de bit-rij van interval f_n om de bit-rij q_{n+1} te vormen. Daarom worden toegevoegde bits per stap gelijk in de uitvoer van het resultaat geplaatst. Tenslotte, bij het stopsymbool, voegt men aan f_n de laatste bits toe zodat de breuk $q = 0, f$ volledig door de laatst gevonden partitie wordt ingesloten.

Het decompressie-algoritme van Elias maakt op gelijksoortige wijze gebruik van de berekening van de frequentieverhoudingen $F_n(a_i)$ en de afmetingsverhoudingen $p_n(a_i)$ om tijdens stap n vanuit bit-rij q_n de partitie a_i te bepalen en symbool a_i te genereren.

2.8 Niet-verliesvrije compressie

Een van de consequenties van de informatietheorie van Shannon is de mogelijkheid compressiemethoden te verwezenlijken waarbij de gedecomprimeerde informatie niet meer volledig overeenkomt met de originele informatie.

Als men de statistische eigenschappen van de bron kent, kan men een functie $R(D)$ vinden met het *algoritme van Blahut*. De functie geeft de relatie weer tussen informatie van de gedecomprimeerde rij en de mate van vervorming (*distorsie*) D. Indien vervorming onacceptabel is $D = 0$, dan is $R(0)$ gelijk aan de entropie van de bron H:



Figuur 2.5: De distorsiefunctie

Het gaat voor deze cursus te ver om de theoretische achtergronden van de compressie met verlies te behandelen. Wel moet opgemerkt worden dat compressie met verlies (‘*lossy compression*’) enkele belangrijke toepassingen kent zoals *JPEG* (Joint Photographic Expert Group), *MPEG* (Moving Picture Expert Group) etc.

Bij beeldinformatie wordt bijvoorbeeld gebruik gemaakt van het feit dat ons oog veel gevoeliger is voor variatie van intensiteit dan voor variatie van kleur. Een beeld in het RGB-systeem (informatie over de absolute rood-, groen- en blauw intensiteit) wordt middels een transformatie omgevormd tot het YIQ-systeem (Y is zwart/wit informatie, I en Q zijn kleurcomponenten). Op de I en Q componenten is een redelijke hoeveelheid informatieverlies acceptabel zonder merkbaar kwaliteitsverlies. Op deze manier kan een *compressieverhouding* gerealiseerd worden die tussen de 40% en 80% ligt.

Een andere methode voor beeldcompressie, de *vector quantisatie* (*VQ*), bestaat uit blokken aaneengesloten pixels (2×2, 3×3 $\ldots$ $n \times n$) voor hun eigenschappen uit te middelen. Alle pixels uit zo’n blok krijgen dan dezelfde gemiddelde eigenschappen. Soms worden alleen de hoekpixels uit zo’n $n \times n$ blok gebruikt, waaruit het decompressie-algoritme de tussengelegen pixels kan interpoleren. Andere ‘lossy compressiemethoden’ voor beeldopslag maken gebruik van een volledig opgebouwd startframe, waarbij alleen de verschillen (‘*deltacompressie*’) naar de volgende frames worden toegevoegd.

Bij lossy-JPEG wordt gebruik gemaakt van *DCT* (*Discrete Cosine Transform*), een bijzondere vorm van *discrete Fouriertransformatie*[1]. De pixels worden gegroepeerd in 8×8 blokken. Van elk blok wordt door DCT de gemiddelde waarde en de trage tot snelle beeldovergangen berekend. Dit heeft het voordeel dat je van de beeldinformatie in een groep pixels beter de snelle overgangen kan kwijtraken dan de minder snelle overgangen of de gemiddelde informatie. Bij de lossy-JPEG methode is altijd sprake van informatieverlies door afbreek- en afrondingsfouten bij de DCT berekeningen.

Audiocompressie zoals *mp3* (*MPEG1/2 layer 3*) maakt gebruik van zintuiglijke effecten. De meeste mensen kunnen alleen toonhoogten tussen de 400 en 20000 Hz waarnemen. Een paar volwassenen en de meeste zuigelingen hebben een absoluut gehoor. Zij blijken in staat te zijn een toonhoogte te herkennen, zonder gebruik te maken van een andere toonhoogte. Het menselijk gehoor blijkt voor geluidsintensiteit een logaritmisch gedrag te vertonen. Bovendien wordt ons gehoor zeer sterk beïnvloed door een verandering van geluidsintensiteit. Bijvoorbeeld, na stilte wordt een geluidsignaal sterker waargenomen dan na herrie. Het omgekeerde effect treedt ook op, na herrie wordt een geluidsignaal zwakker waargenomen dan na stilte. Dit verdringingseffect is mede afhankelijk van de toonhoogte van het geluidssignaal. Door gebruik te maken van het verdringingseffect van zachte geluiden, kan de compressieverhouding bij audiosignalen uiteindelijk onder de 10% komen.

2.9 Opgaven

1. Een bron genereert een onafhankelijke rij symbolen uit het alfabet $\{0,1\}$. De kans op een 0 is $0,9$, de kans op een 1 is $0,1$. De rij getallen wordt met een ‘Run-Length codering’ gecodeerd tot een rij met symbolen uit het alfabet $\{a,b,c,d,e,f,g,h,i\}$. Tenslotte worden deze ‘run-length’ codewoorden weer gecodeerd met een Huffmancode.

b-code	r-code
1	a
01	b
001	c
0001	d
00001	e
000001	f
0000001	g
00000001	h
00000000	i

   (a) Bereken de entropie van de bron.

[1] Cursus Signaalverwerking

(b) Bereken het gemiddeld aantal bronsymbolen per run-length codewoord.

(c) Bereken het gemiddeld aantal bit van de Huffmancode per run-length woord.

(d) Bereken het gemiddelde aantal bronsymbolen b per Huffmansymbool h.

2. Hoeveel procent van de rijen van n letters uit grote alfabetten van $2 < m$ lettersymbolen, zouden er bij grote bestanden ($n \to \infty$) niet verlies-vrij gecomprimeerd kunnen worden?

3. In 1820 ontwikkelde Louis Braille een code die blinden kunnen waarnemen door in dik papier de aanwezigheid of afwezigheid van puntvormige verdikkingen te voelen. Elk symbool bestaat uit 3×2 posities, waarmee men $2^6 = 64$ symbolen kan maken. De posities zijn als volgt genummerd:

1	4
2	5
3	6

Hoe zou u de letters van het braille-alfabet ontwerpen?

4. Waarom zouden de makers van '*gzip*' de blokverwijzingen beperkt hebben?

5. In het DNA van de bacterie *Micrococcus Lysodeiktus* hebben de basen A, C, T, G de volgende waarschijnlijkheid: $P(A) = P(T) = 29/200$ en $P(C) = P(G) = 71/200$. Bij de bacterie *E. Coli* is deze verdeling: $P(A) = P(T) = P(C) = P(G) = 1/4$

Welke bacterie zou van de twee het meest complexe organisme zijn?

6. Het decompressie-algoritme van het programma '*bzip2*' maakt gebruik van een inverse permutatie $T(i)$ om uit de laatste kolom de bronrij terug te vinden. Wat is er fout aan de volgende redenatie om de bronrij terug te vinden uit tabel 2.5, gegeven dat de rij-index van de bronrij de waarde 3 heeft?

> Men kan uit de tabel 2.5 aflezen dat de onbekende bronrij moet beginnen met een '*o*' (rij 3 , kolom 2) en eindigen met een '*b*' (rij 3 kolom 1). Vervolgens blijkt de rij bronsymbolen uit alle andere tweetallen te zijn opgebouwd voor de tussenliggende symbolen: '*ob*', '*ro*', '*oo*', '*rr*', '*or*'. Wij weten alleen niet in welke volgorde. Wel is bekend dat elk tweetal exact één keer gebruikt moet worden.
>
> Met deze gegevens kunnen wij de bronrij herstellen: Start met '*o*' dan zijn er twee volgende letters mogelijk '*b*' en '*o*' (vanwege '*ob*' en '*oo*'). De rij '*ob*' heeft geen opvolger en is dus geen oplossing. De rij '*oo*' heeft als uitbreiding '*oor*' (vanwege '*or*'). Dit passen en meten kunnen wij herhalen tot wij de originele bronrij '*oorrob*' gevonden hebben.

7. De berekening van de afmetingsverhoudingen bij arithmetic codering is gebaseerd op het theorema van *Bayes* en de regel van *Laplace* om in 'one-pass' te convergeren naar de echte relatieve frequenties:

$$P(x_n = a_i \mid x_1, x_2, \ldots, x_{n-1}) = \frac{1 + F_n(a_i)}{\sum_{j=1}^{m}(1 + F_n(a_j))} = p_n(a_i)$$

Waarbij $F_n(a_i)$ het aantal keren is dat symbool a_i in de rij $x_1, x_2, \ldots, x_{n-1}$ is opgetreden. Als gegeven is dat het alfabet bestaat uit de symbolen *a*, *b* en het stopsymbool •, kunnen wij in principe bij een gegeven string de afmetingsverhoudingen berekenen:

$$\begin{aligned} P(abbb\bullet) &= P(\bullet \mid abbb) \cdot P(abbb) \\ &= P(\bullet \mid abbb) \cdot P(b \mid abb) \cdot P(abb) \\ &= P(\bullet \mid abbb) \cdot P(b \mid abb) \cdot P(b \mid ab) \cdot P(ab) \\ &= P(\bullet \mid abbb) \cdot P(b \mid abb) \cdot P(b \mid ab) \cdot P(b \mid a) \cdot P(a) \end{aligned}$$

Bereken de afmetingsverhoudingen '*abbb*•' voor de volgenden 5 stappen en geef de successievelijke partitionering aan op een lijnstuk met de lengte van 1:

stap	afmetingsverhoudingen		
1	$P(a) = \ldots$	$P(b) = \ldots$	$P(\bullet) = \ldots$
2	$P(a \mid a) = \ldots$	$P(b \mid a) = \ldots$	$p(\bullet \mid a) = \ldots$
3	$P(a \mid ab) = \ldots$	$P(b \mid ab) = \ldots$	$p(\bullet \mid ab) = \ldots$
4	$P(a \mid abb) = \ldots$	$P(b \mid abb) = \ldots$	$p(\bullet \mid abb) = \ldots$
5	$P(a \mid abbb) = \ldots$	$P(b \mid abbb) = \ldots$	$p(\bullet \mid abbb) = \ldots$

8. Als men een een bron met maximale entropie door een (verliesvrije) decompressie algoritme haalt, wat is dan de entropie van de het resultaat?

Hoofdstuk 3

Coderingen voor storingsrijke omgevingen

Redundantie is noodzakelijk indien het bericht op een of andere manier verminkt kan worden. De overvloedigheid van de informatie stelt ons in staat om het originele bericht te reconstrueren.

Voorbeeld 3.1

Stel dat een bericht 3 keer herhaald wordt, dan is de relatieve redundantie $\frac{R(X)}{H_{max}(X)} = \frac{3-1}{3} = 66,7\%$. Indien een van de drie berichten verminkt wordt, is het meestal mogelijk de originele informatie terug te vinden.

Voorbeeld 3.2

In de verminkte zin 'LPN N FTSN ZN GZND SPRTN' valt na wat puzzelen de zin 'LOPEN EN FIETSEN ZIJN GEZONDE SPORTEN' te herkennen. In de Nederlandse schrijftaal zou men zonder verlies van informatie een gedeelte van de klinkers kunnen weglaten (zoals in het Hebreeuws). De relatieve redundantie, zonder rekening te houden met afhankelijkheden tussen karakters, is ongeveer 50%. Indien wij rekening houden met afhankelijkheden van hogere orde, blijkt dat het mogelijk is om uit ongeveer $\frac{1}{5}$ van een Nederlands bericht het origineel te reconstrueren. Dit komt overeen met een maximale relatieve redundantie van 80%.

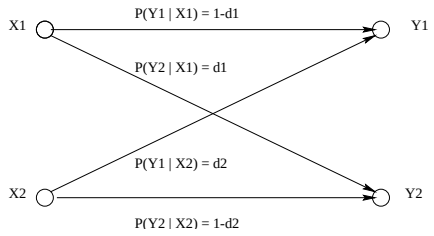
In dit hoofdstuk zullen wij methoden van het detecteren (opsporen) en corrigeren van verminkingen bespreken. Om het effect van verminking te begrijpen zullen wij eerst de werking van een *kanaal* bestuderen.

3.1 Het kanaalcoderingstheorema van Shannon

In de informatietheorie staat het (abstracte) begrip kanaal voor een transmissiekanaal of een opslagmedium. In een kanaal kan door storingen verminking van informatie plaats vinden.

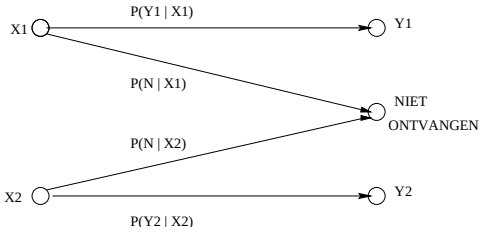
De stochastische variabelen X en Y zijn respectievelijk invoer- en uitvoervariabelen. Stel dat wij van een kanaal de samengestelde frequentieverdeling gemeten hebben. Deze relatieve frequenties worden in een tabel $P(X,Y)$ genoteerd, waarbij de rij- en de kolomsommen de marginale kansverdelingen $P(Y)$ en $P(X)$ voorstellen. Uit deze tabel kunnen wij met de formule van *Bayes* de equivocatietabel $P(Y|X) = P(X,Y)/P(X)$ bepalen. Laten wij bijvoorbeeld het binaire *asymmetrische kanaal* met de volgende equivocatietabel nemen:

$P(Y\|X)$	x_1	x_2
y_1	$P(Y=y_1\|X=x_1)=1-d_1$	$P(Y=y_1\|X=x_2)=d_2$
y_2	$P(Y=y_2\|X=x_1)=d_1$	$P(Y=y_2\|X=x_2)=1-d_2$



Figuur 3.1: Binair asymmetrisch kanaal

Een storing kan de informatie zodanig verminken dat een symbool verandert in een ander symbool. Soms kunnen verminkte symbolen onherkenbaar worden (zie figuur 3.2):



Figuur 3.2: Binair kanaal met verdwijnende symbolen

Het is ook mogelijk dat er een *geheugenwerking* bestaat tussen de codewoorden. Een codewoord kan invloed hebben op een volgend codewoord. Wij zullen ons beperken tot

bron- en kanaalmodellen zonder geheugenwerking. Welke informatie gaat over een kanaal? Om deze vraag te beantwoorden moeten wij gebruik maken van het begrip '*onderlinge entropie*' $H(X;Y)$. Indien de onderlinge entropie nul is, zijn de variabelen X en Y totaal onafhankelijk. Bij de hoogste mate van afhankelijkheid is de onderlinge entropie gelijk aan de laagste entropie van de twee variabelen $H(X;Y) = min(H(X), H(Y))$.

In de informatietheorie zijn er veel modellen voor kanalen. Een van de meest gebruikelijke modellen is het *binaire symmetrische kanaal* (*BSC*). Het BSC heeft de eigenschap dat elk ingangssymbool evenveel kans ($d = d_1 = d_2$) heeft om verminkt te worden:

$P(Y\|X)$	$P(X = x_1) = p$	$P(X = x_2) = 1 - p$
y_1	$P(Y = y_1\|X = x_1) = 1 - d$	$P(Y = y_1\|X = x_2) = d$
y_2	$P(Y = y_2\|X = x_1) = d$	$P(Y = y_2\|X = x_2) = 1 - d$

Wij kunnen nu van een binair symmetrisch kanaal met $0 < d \leq 0,5$ de onderlinge entropie bepalen. Deze onderlinge entropie is een maat van afhankelijkheid tussen X en Y. Hoe lager deze onderlinge entropie, des te onafhankelijker de waarden X en Y, des te lager de *informatieoverdracht*. Nu is deze informatieoverdracht in ons voorbeeld afhankelijk van de eigenschap van de bron X met de kansverdeling $p, 1-p$ en de foutkans d, de *Binary Error Rate*. Alleen de foutkans d is een echte eigenschap van het kanaal. Een definitie van de *discrete kanaalcapaciteit* mag dus alleen afhankelijk zijn van de eigenschappen van het kanaal. Daarom definieerde Shannon de discrete kanaalcapaciteit C als de maximale informatieoverdracht, dit is de maximale onderlinge entropie die mogelijk is in dat kanaal:

$$C = max(H(X;Y)) \quad \text{[bit]} \tag{3.1}$$

Men kan deze maximale onderlinge entropie beschouwen als de gemiddelde informatieinhoud van een transmissiecodewoord. Deze maximale onderlinge entropie geldt voor alle mogelijke verdelingen $\{p_1, p_2, \ldots p_n\}$ van de broncode X. Het is dus mogelijk bij een gegeven kanaal met een foutkans d een broncodering te vinden waarbij de discrete kanaalcapaciteit zo groot mogelijk is en de kans op verminkingen binnen een vastgestelde grens blijft.

> **Kanaalcoderingstheorema:** *"Voor een kanaal met eindige capaciteit C bestaat bij een voldoende lengte L van de broncode X een verdeling van bronsymbolen zodanig dat de kans op een fout kleiner is dan een van te voren vastgestelde mate."* (C. Shannon)

Voor het binair symmetrisch kanaal geldt:

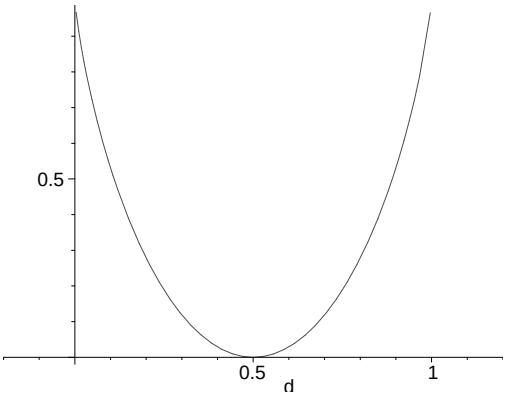
$$\begin{aligned} H(Y) &= -P(Y = y_1) \cdot \mathrm{ld}\,(P(Y = y_1)) - P(Y = y_2) \cdot \mathrm{ld}\,(P(Y = y_2)) \\ &= -(p \cdot (1-d) + (1-p) \cdot d)\mathrm{ld}\,(p \cdot (1-d) + (1-p) \cdot d) \\ &\quad -(p \cdot d + (1-p) \cdot (1-d))\mathrm{ld}\,(p \cdot d + (1-p) \cdot (1-d)) \\ &\leq 1 \quad \text{als} \quad p \cdot (1-d) + (1-p) \cdot d = \tfrac{1}{2} \end{aligned} \tag{3.2}$$

Met behulp van formule 1.12 wordt de equivocatie $H(Y|X)$ berekend:

$$H(Y|X) = -p \cdot H(Y|X = x_1) + (1-p) \cdot H(Y|X = x_2) = -d \cdot \mathrm{ld}(d) - (1-d) \cdot \mathrm{ld}(1-d) \tag{3.3}$$

De discrete kanaalcapaciteit van een BSC wordt:

$$C = \overbrace{max\underbrace{(H(Y) - H(Y|X))}_{H(X;Y)}}^{\text{voor alle verdelingen } P(X=x_i)=p_i} = \underbrace{H(Y)}_{p_1=p_2=\frac{1}{2}} - H(Y|X) = 1 + d \cdot \mathrm{ld}(d) + (1-d) \cdot \mathrm{ld}(1-d) \tag{3.4}$$



Figuur 3.3: De capaciteit C van een BSC als functie van de foutkans d.

Omdat de eigenschappen van het kanaal gegeven zijn, wordt gegeven een foutkans d (zie figuur 3.3) de capaciteit $C(d)$ van een BSC bereikt als $p \cdot (1-d) + (1-p) \cdot d = \frac{1}{2}$. Met andere woorden, als $d \neq \frac{1}{2}$ dan volgt daaruit dat de broncode X uniform verdeeld moet zijn ($p_{1,2} = \frac{1}{2}$) om geen capaciteitsverlies te krijgen.

3.2 Decodeerprincipes

Het decoderen van codewoorden naar bronwoorden in een storingsrijke omgeving leidt tot de volgende vraag: Welk bronwoord x_i moeten wij bij een gegeven codewoord y_j aannemen? Het antwoord op deze vraag wordt gegeven door de volgende decodeerprincipes:

Maximum-a-Posteriori (MAP) decodering: Kies x_i zodanig dat $P(X = x_i|Y = y_j)$ maximaal is. De decoder kiest dus het bronwoord x_i dat gegeven het codewoord y_j het meest waarschijnlijk is. MAP wordt gebruikt om de gemiddelde decodeerfout zo laag mogelijk te houden;

Maximum Likelihood (ML) decodering: Kies x_i zodanig dat $P(Y = y_j|X = x_i)$ maximaal is. De decoder kiest het bronwoord x_i dat het codewoord y_j het meest waarschijnlijk maakt. ML wordt gebruikt als de bronwoorden uniform verdeeld zijn;

Minimum Distance (MD) decodering: Kies x_i zodanig dat afstand met y_j minimaal is. De decoder kiest het bronwoord x_i waarvan het codewoord y_i het minst aantal symbolen afwijkt van het codewoord y_j. MD wordt gebruikt als er genoeg redundantie aanwezig is om voldoende afstand tussen de codewoorden te creëren.

Wij zullen het verschil tussen MAP en ML demonstreren aan de hand van het BSC. Daarvoor moeten wij eerst uit de conditionele kansen $P(Y|X)$

$P(Y\|X)$	x_1	x_2
y_1	$P(Y = y_1\|X = x_1) = 1-d$	$P(Y = y_1\|X = x_2) = d$
y_2	$P(Y = y_2\|X = x_1) = d$	$P(Y = y_2\|X = x_2) = 1-d$

de kansen $P(X|Y)$ berekenen, met de omkeerformule van Bayes. Uit deze formule en de symmetrie eigenschap van het BSC volgt:

$$P(X = x_1|Y = y_1) = \frac{P(Y = y_1|X = x_1) \cdot P(X = x_1)}{P(Y = y_1|X = x_1) \cdot P(X = x_1) + P(Y = y_1|X = x_2) \cdot P(X = x_2)}$$

Hieruit volgt op dezelfde manier voor alle andere conditionele kansen:

$P(X\|Y)$	y_1	y_2
x_1	$P(X = x_1\|Y = y_1) = \frac{p\cdot(1-d)}{p+d-2\cdot d\cdot p}$	$P(X = x_1\|Y = y_2) = \frac{p\cdot d}{1-d-p+2\cdot d\cdot p}$
x_2	$P(X = x_2\|Y = y_1) = \frac{(1-p)\cdot d}{p+d-2\cdot d\cdot p}$	$P(X = x_2\|Y = y_2) = \frac{(1-p)\cdot(1-d)}{1-d-p+2\cdot d\cdot p}$

Opmerking: Voor het storingsvrije BSC ($d = 0$) blijken de kansen $P(Y = y_1|X = x_1) = P(Y = y_2|X = x_2) = 1$ en de kansen $P(Y = y_2|X = x_1) = P(Y = y_1|X = x_2) = 0$. In het geval dat $d = 1$ gedraagt het BSC zich, vreemd genoeg, ook als een storingsvrij kanaal, de kansen $P(Y = y_1|X = x_1) = P(Y = y_2|X = x_2) = 0$ en de kansen $P(Y = y_2|X = x_1) = P(Y = y_1|X = x_2) = 1$. Het lijkt alsof het BSC $d = 1$ een *inverterende werking* krijgt. Het symbool x_1 wordt altijd y_2 en het symbool x_2 wordt altijd y_1. Het BSC geeft echter de meeste kans op verminking als $d = 0,5$. Dit blijkt overeen te komen met capaciteit als functie van de foutkans (zie figuur 3.3).

Bij een gegeven y_1 kiest men bij het MAP principe voor:

$$max(\overbrace{\frac{p\cdot(1-d)}{p+d-2\cdot d\cdot p}}^{x_1}, \overbrace{\frac{(1-p)\cdot d}{p+d-2\cdot d\cdot p}}^{x_2}) \tag{3.5}$$

Bij niet-uniform verdeelde broncodes is de verdeling van de broncode belangrijk bij de reconstructie van de verminkte informatie. Zodra de broncode uniform verdeeld is ($p\frac{1}{2}$), gaat formule 3.5 over in formule 3.6. Daarom kiest men bij uniform verdeelde broncode voor het ML principe. Bij dit decodeerprincipe speelt de broncodeverdeling geen rol meer:

$$max(\overbrace{1-d}^{x_1}, \overbrace{d}^{x_2}) \tag{3.6}$$

Het derde principe, het MD principe, is gebaseerd op het begrip afstand tussen codewoorden. Er moet voldoende redundantie zijn om een minimale afstand tussen de codewoorden te detecteren. Een voorbeeld van een MD-decoder is de eerste-orde *Hammingcode*. Om de Hammingcode te bespreken moeten wij eerst het begrip *Hamming Distance* behandelen dat gebaseerd is op redundantie in de codewoorden.

3.2.1 Hamming Distance

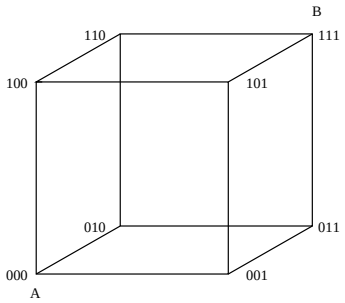
Zoals bij het MD-principe gesteld werd, kan bij voldoende redundantie, de afstand tussen de codewoorden vergroot worden. Hoe groter het aantal redundante bits per codewoord, hoe kleiner de kans dat een verminking het codewoord verandert in een ander geldig codewoord zonder dat dit gedetecteerd wordt. Om de afstand tussen de codewoorden te definieren, gebruikt men de *Hamming Distance*. De Hamming Distance is het aantal verschillen tussen bits op overeenkomstige posities van twee codewoorden. Wij gaan ervan uit dat deze codewoorden allemaal dezelfde lengte L hebben. De verschillen tussen bits op de overeenkomstige bitposities wordt met een $\oplus$ operatie op de twee codewoorden x_i en x_j bepaald, de afstand $d(x_i,x_j)$ is gedefinieerd als $||x_i \oplus x_j||$, het aantal bits dat 1 is. Bijvoorbeeld de afstand tussen het codewoord 0100101 en het codewoord 01100110 is gelijk aan:

$$d(x_i,x_j) = ||0100101 \oplus 0110110|| = ||\overbrace{0010011}^{3\times 1}|| = 3$$

Voorbeeld 3.3

Twee codewoorden worden afgebeeld op woorden van 3 bit. Bijvoorbeeld codewoord $A = 000$ en codewoord $B = 111$. Deze codering kan worden voorgesteld als een 3-dimensionale kubus:

Als één bit in een codewoord omklapt, wordt een van naastgelegen codewoorden ontvangen. Dit zijn de hoekpunten die slechts één ribbe van het hoekpunt afstaan. Voor het bronwoord $A = 000$ zijn dit de codewoorden 001, 010 en 100. Voor het bronwoord $B = 111$ zijn dit de codewoorden 110,101 en 011. Alle codewoorden met een 1-bit verminking zijn dan te herkennen als fout. Als er bovendien geen codewoorden zijn die een gemeenschappelijke Hamming Distance van 1 bit hebben tot A en B, is het zelfs mogelijk het originele bronwoord te bepalen.



Figuur 3.4: Een 3-bit Hamming Distance kubus

Zoals uit het voorbeeld blijkt, is er tussen alle gecodeerde bronwoorden, een '*Minimum Hamming Distance*' noodzakelijk. Vanuit deze minimale afstand tussen de codewoorden kunnen wij berekenen hoelang de codewoorden minimaal moeten zijn om een fout te detecteren en te corrigeren.

> **Minimum Hamming Distance theorema:** "*Bij een minimum Hamming Distance MHD is de decoder in staat alle* $N \leq MHD - 1$ *bitfouten te detecteren en alle* $N \leq \frac{MHD-1}{2}$ *bitfouten te corrigeren.*"

Voorbeeld 3.4

> Een codering bestaat uit $00 \rightarrow 00000$, $10 \rightarrow 00111$, $01 \rightarrow 01110$ en $11 \rightarrow 11111$. De minimum Hamming afstand is $MHD = 2$. Hoewel de decodering alle 1-bit fouten kan detecteren, is zij niet in staat tot foutcorrectie.

Indien een code in staat is bitfouten te corrigeren noemt men haar een *fouttolerante code*. Hoeveel redundantie is nodig voor fouttolerantie? Om deze vraag te beantwoorden nemen wij bronwoorden met een vaste lengte van L bit. Dit betekent dat er maximaal 2^L bronwoorden moeten worden afgebeeld op een verzameling codewoorden van ieder K bit. Wij kunnen dit vergelijken met het plaatsen van 2^L hoekpunten op een hyperkubus met 2^K hoeken. Elk codewoord heeft in deze hyperkubus K directe buurpunten. Per codewoord moeten dus $1+K$ hoeken in de hyperkubus gereserveerd worden. Daaruit volgt dat met K bit maximaal $\frac{2^K}{1+K}$ codewoorden mogelijk zijn indien zij ongevoelig voor één-bit fouten moeten zijn. Hieruit volgt dat bij bronwoorden van L bit, de codewoordlengte K aan de volgende eis moet voldoen:

$$2^L \leq \frac{2^K}{1+K} \qquad \text{(tolerant voor 1-bit fouten)}$$

De redundantie van deze code komt daarmee op $K-L$ bit. Als wij deze redenatie uitbreiden voor meer-bit fouten dan zijn bij een Hammingcode van K bit $\binom{K}{1}$ één-bits fouten

mogelijk. Het aantal 2-bit fouten is $\binom{K}{2}$. Het aantal N-bit fouten is $\binom{K}{N}$. Per bronwoord moeten in de hyperkubus $1+\binom{K}{1}+\binom{K}{2}+\ldots+\binom{K}{N}$ hoekpunten gereserveerd worden om N-bit fouten te detecteren en te corrigeren. Hieruit volgt bij N-bit fouten of minder, bronwoorden van L bits en codewoorden van K-bit aan de volgende eis moeten voldoen:

$$2^L \leq \frac{2^K}{1+\binom{K}{1}+\binom{K}{2}+\ldots+\binom{K}{N}} \qquad \text{(tolerant voor N-bits fouten)}$$

Voorbeeld 3.5

Bijvoorbeeld 4 bronwoorden ($L = 2$ bit) kunnen met codewoorden van $K = 7$ bits tolerant worden voor 2-bit fouten:

$$2^2 \leq \frac{2^7}{1+\binom{7}{1}+\binom{7}{2}} = 4,4$$

De redundantie per codewoord $K - L = 5$ bit

3.2.2 Pariteitscontrole

Een veel gebruikte foutdetectie methode is de pariteitscontrole. Met de pariteitscontrole kan een oneven aantal bitfouten gedetecteerd worden in een woord van K-bits. Aan het codewoord van K-bits wordt 1 extra bit redundantie toegevoegd, het *pariteitsbit*. Dit pariteitsbit is een $\oplus$ operatie op alle K-bits van het codewoord. Er zijn twee varianten van de pariteit, een even- (*even-parity*) en een oneven pariteit (*odd-parity*). Een even pariteit is 'waar' als het aantal 1-bits in het codewoord even is. Oneven pariteit is 'waar' als het aantal 1-bits oneven is. Voor de pariteitscontrole maakt het niet uit of de pariteit op een even of oneven aantal 1-bits gebaseerd is, als de pariteit bij het decodeerproces maar overeenkomt met de pariteit bij het encodeerproces. Om de pariteit te bepalen is een functie noodzakelijk die het aantal 1 bits in een woord telt, de gewichtsfunctie. Als een bit 0 waarde heeft, dan geeft de gewichtsfunctie de decimale waarde 0, anders de decimale waarde 1:

$$\textbf{if } d = 1 \textbf{ then } ||d|| = 1 \textbf{ else } ||d|| = 0$$

De gewichtsfunctie werkt ook op codewoorden van K-bits:

$$||d_1 d_2 \ldots d_n|| = ||d_1|| + ||d_2|| + \ldots + ||d_K||$$

Bijvoorbeeld:

$$||10010110|| = 4$$

Pariteitscontrole werkt alleen als een oneven aantal bits verminkt is. Een even aantal verminkte bits wordt niet gedetecteerd. Ondanks deze ernstige beperking van de pariteitscontrole, wordt zij veel bij datatransmissie over korte afstanden gebruikt. Meestal worden codewoorden van 8 bits gevolgd door 1 pariteitsbit. Men noemt dit *horizontale pariteitscontrole*.

Omdat een horizontale pariteitscontrole niet in staat is om de positie van de fout in een codewoord aan te geven, is voor foutcorrectie meer redundantie nodig. Bij een vast aantal codewoorden met een gelijke lengte K, een *blok*, is het mogelijk een combinatie van *horizontale pariteitscontrole* en *verticale pariteitscontrole* toe te passen. Net zoals horizontale pariteit beschouwd kan worden als een $\oplus$ operatie op de rij bits binnen een codewoord, is de verticale pariteitscontrole een $\oplus$ kolomoperatie op de codewoorden in het blok. Deze verticale pariteitscontrole wordt *longitudinale pariteitscontrole* (LPC) of *checksum* genoemd.

woord	data bits								hor. par.	
1	d	d	d	d	d	d	d	d	p	
2	d	d	d	d	d	d	d	d	p	
3	d	d	d	.	d	d	d	d	f	← foute rij
4	d	d	d	d	d	d	d	d	p	
5	d	d	d	d	d	d	d	d	p	
6	d	d	d	d	d	d	d	d	p	
7	d	d	d	d	d	d	d	d	p	
8	d	d	d	d	d	d	d	d	p	
9	p	p	p	f	p	p	p	p	p	vert. par.
				↑						foute kolom

Bij elektronische transmissie over lange afstanden waarbij meerdere bits naast elkaar in een codewoord verminkt kunnen worden, zogenaamde ('*bursts*'), is deze combinatie van horizontale- en verticale pariteitscontrole niet geschikt. Daarom wordt LPC alleen gebruikt bij datatransmissie over korte afstanden.

3.2.3 CRC

Het principe van '*Cyclische Redundantie Code*' (*CRC*) kunnen wij aan de hand van een eenvoudig rekenkundig voorbeeld uitleggen:

Voorbeeld 3.6

De deelbaarheid van een getal door 9 is zeer gemakkelijk te bepalen: Een getal is deelbaar door 9 als de som van de decimalen deelbaar is door 9. Bijvoorbeeld, het getal 32913 is deelbaar door 9 omdat $3+2+9+1+3=18=1+8=9$ deelbaar is door 9. Dit getal 9 noemen wij het '*generator-getal*' G.

Dit effect kunnen wij gebruiken als wij een getal M willen verzenden via een onbetrouwbaar kanaal. Wij coderen het getal M in een nieuw getal T zodanig dat het

getal T deelbaar moet zijn door het generator-getal G. Indien tijdens de transmissie een verminking E heeft plaats gevonden, zal de ontvanger dat in $G-1$ van de G gevallen kunnen ontdekken omdat het ontvangen getal $T+E$ niet meer deelbaar is door G. De codering van getal M in een getal T geschiedt op de volgende manier:

1. Deel het getal T geheeltallig door G en bepaal de rest R. Wiskundig schrijven wij dat als: $(R = M) \bmod G$. Bijvoorbeeld $(R = 2 = 45182) \bmod 9$;
2. Om een getal T maken dat deelbaar is door 9, moeten wij van het getal M de rest R aftrekken. Dit gebeurt door een extra decimaal met de waarde met de negatieve waarde $(-R) \bmod 9$ toe te voegen. In ons voorbeeld is dit: $(7 = -2) \bmod 9$;
3. Het getal T wordt: $M \times 10 + ((-R) \bmod 9)$. In ons voorbeeld: $T = 451827$. Het getal T is nu deelbaar door 9, wiskundig: $(T = 0) \bmod 9$.

Het CRC-principe beschouwt een bericht als een groot getal. Dit getal, op zich weer te beschouwen als een groot binair getal, wordt geheeltallig gedeeld door een ander getal. De rest van deze deling wordt aan het getal toegevoegd om het getal met extra redundantie te beschermen tegen transmissiefouten. Nu blijkt bij CRC het deelbaarheidsprincipe iets moeilijker omdat wij gebruik moeten maken van een ander soort rekenkunde. Deze rekenkunde wordt *polynomische rekenkunde* genoemd.

Wij kunnen een rij met 0-en en 1-en weergeven als een polynoom. Bijvoorbeeld de rij 1101 wordt weergegeven door de polynoom $(x^3+x^2+1) \bmod 2$ waarvan de coëfficiënten alleen 0 of 1 kunnen zijn.

De som 101 en 001 wordt $(x^2+1+1=x^2) \bmod 2$, dit komt overeen met $101+001=100$. Het grote verschil met normale rekenkunde is het feit dat er geen sprake is van een overloop (‘*carry*’) van de ene coëfficiënt naar de volgende. Polynomisch vermenigvuldigen en delen en delen gaat op soortgelijke wijze:

$$(x^2+1)(x+1) = x^4+x^2+x^2+1 = x^4+\overbrace{(1+1)}^{0}x^2+1 = x^4+1$$

Voorbeeld 3.7

$$\begin{array}{r} x^3+x^2 \\ x\ +1 \\ \hline x^3+x^2 \\ x^4+x^3 \\ \hline x^4\quad +x^2 \end{array}$$

Het resultaat van x^3+x^2 vermenigvuldigd met $x+1$ is x^4+x^2.

Een bijzondere eigenschap heeft het polynomisch vermenigvuldigen met x^p. Dit geeft een verschuiving naar links met p bit, aangevuld met ‘0’-en. Bijvoorbeeld als wij $x^5+x^4+x^2$ (binair 1101), vermenigvuldigen met x^2 (binair 10) dan geeft dit $(x^5+x^4+x^2)\cdot x^2 = x^7+x^6+x^4$ (binair 110100).

Voorbeeld 3.8

$$
\begin{array}{l|l}
x+1 & x^4 \quad +x^2+1 \;\big|\; x^3+x^2 \\
\cline{2-2}
 & \underline{x^4+x^3} \\
 & \quad x^3+x^2 \\
 & \quad \underline{x^3+x^2} \\
 & \qquad\quad +1
\end{array}
$$

Het resultaat van x^4+x^2+1 gedeeld door $x+1$ is x^3+x^2 rest 1.

Het vermenigvuldigen, het ontbinden in factoren, het bepalen van de rest van een deling van polynomen wordt door *Maple* ondersteund:

```
> expand((x^2+x+1)*(x+1)) mod 2;
```

$$x^3+1$$

```
> Factor(x^3+1) mod 2;
```

$$(x^2+x+1)(x+1)$$

```
> Rem(x^4+x^2+1,x+1,x) mod 2;
```

$$1$$

Het CRC-algoritme zal, met gebruikmaking van de '*generator-polynoom*' $G(x)$ met de *graad r* (de macht van hoogste term van $G(x)$), het bronbericht $M(x)$ coderen tot het codebericht $T(x)$.

$$rest\left(\frac{x^r \cdot M(x)}{G(x)}\right) = R(x)$$

De polynomische correctie $-R(x)$ wordt aan het brongetal $M(x)$ als $r-1$ bit redundantie 'toegevoegd' om de code $T(x)$ te vormen:

$$(T(x) = x^r \cdot M(x) - R(x) = x^r \cdot M(x) + R(x)) \bmod 2$$

$T(x)$ is de code die verzonden wordt. De rest van de deling $T(x)/G(x)$ moet bij ontvangst 0 zijn, behalve als een fout $E(x)$ is op getreden:

$$rest\left(\frac{T(x)+E(x)}{G(x)}\right) \neq 0$$

Welke generator-polynoom $G(x)$ komt in aanmerking? De graad r geeft aan hoeveel redundantie in het bericht $T(x)$ wordt toegevoegd. De kans om een fout te detecteren is afhankelijk van deze redundantie en de lengte van de bronrij $M(x)$. Om een idee te krijgen hoe een geschikte generator-polynoom gekozen moet worden, moeten wij naar het type van de fouten kijken dat wij willen detecteren. Fouten $E(x)$ die een veelvoud zijn van $G(x)$ zijn in principe niet te detecteren omdat anders $rest((T(x)+E(x))/G(x)) = rest(E(x)/G(x)) = 0$ geldt.

Één-bit fouten: Dit zijn fouten waarin maar één bit fout is, $E(x) = 1\overbrace{0\ldots0}^{p} = x^p$. Als $G(x) = x^q \leq x^p$ één macht heeft dan zal $E(x)$ altijd een meervoud zijn van $G(x)$. Daarom moet $G(x)$ tenminste twee machten of 1-coëfficiënten hebben;

Twee-bit fouten: Dit zijn fouten waar exact twee-bit fout zijn: $E(x) = 10\ldots010\ldots0$. De polynoom wordt beschreven als: $E = x^p + x^q$ waarbij $p > q$. Zij kan worden ontbonden in $E(x) = x^q(x^{p-q}+1)$. Algebraïsch blijken fouten van het type $E = x^{i\cdot 2^j}+1$ ontbonden te kunnen worden in $(x^i+1)^{2^j}$ Als $G(x)$ niet gelijk is aan $(x^i+1)^j \quad (i,j \geq 1)$ dan kunnen deze $E(x)$ geen veelvoud zijn en worden alle twee-bit fouten ontdekt;

Bursts: Dit zijn fouten waar n bits naast elkaar fout zijn, $E(x) = \overbrace{1\ldots1}^{n}\overbrace{0\ldots0}^{p}$. De waarde E kan herschreven worden als een product $E = x^p \cdot (x^{n-1}+x^{n-1}+\ldots+1)$. Als de coëfficiënt van x^0 van $G(x)$ de waarde 1 heeft, kan de verschuivingsfactor x^p van $E(x)$ geen veelvoud zijn van $G(x)$. Ondeelbaarheid van $E(x)$ is nu alleen gegarandeerd als de graad $n-1$ van $x^{n-1}+x^{n-2}+\ldots+1$ kleiner is dan die van $G(x)$. Het is dan mogelijk alle n-bit '*bursts*' te detecteren als n kleiner of gelijk is aan r. Bursts met n langer dan r worden met de kans $0,5^{r-1}$ herkend;

Oneven-aantal-bits fouten: Een oneven aantal bits in $E(x)$ is te detecteren als $G(x)$ een factor $x+1$ heeft. Polynomen $E(x)$ met een oneven aantal 1-coëfficiënten zijn niet-deelbaar door $(x+1)$. Dit kunnen wij bewijzen door aan te nemen dat zo'n polynoom $E(x)$ wel een $x+1$ factor heeft. Wij factoriseren $E(x)$ in $x+1$ en een factor $F(x)$. Hieruit volgt:

$$E(x=1) = (\overbrace{x+1}^{1+1=0})\cdot F(x=1) = 0$$

Dit is tegenstrijdig met $E(x=1)=1$, omdat $E(x)$ een oneven aantal 1-coëfficiënten heeft. Hiermee is bewezen dat een polynoom $E(x)$ met een oneven aantal coëfficiënten

niet deelbaar is door een generatorpolynoom $G(x)$ met een factor $(x+1)$. Niet alle bekende generator-polynomen voldoen aan deze eis.

Uit bovenstaande criteria blijkt dat de 'omkering' van een geschikte generator-polynoom ook geschikt is ('*CRC reversed*'). Hieronder volgt een lijst van veel toegepaste generator-polynomen (zie bijlage: E):

graad r	$G(x)$	naam	toepassing
4	11101_2	CRC-4	
6	1100011_2	CRC-6	
8	100000111_2	CRC-8	
12	1100000001111_2	CRC-12	
15	1100010110011001_2	CRC-15	
16	10001000000100001_2	X25 standard	CRC-CCITT, SDLC/HDLC
16	10000100000010001_2	X25-reversed	
16	11000000000100001_2	CRC-16	CCITT V41, RS232C
16	11000000000000101_2	CRC-16 standard	IBM, ARC
16	10100000000000011_2	CRC-16 reversed	LHA
32	$104C11DB7_{16}$	CRC-32	PKZIP, bzip2, gzip

Tabel 3.1: Veel gebruikte CRC polynomen.

Het blijkt dat de graad m van het bericht $M(x)$ en de graad r van de redundantie $R(x)$ niet van invloed zijn op de detectie van 1- en 2-bit fouten. Toch wordt de kans op het niet herkennen van fouten groter als de berichten langer- of de generatorpolynomen korter worden:

Elke generatorpolynoom $G(X)$ van de graad r heeft 2^r mogelijke resten $M(x)/G(x) = R(x)$. Een bronwoord $M(x)$ van de graad m kan 2^m mogelijke berichten bevatten. Er zijn dus 2^{m-r} berichten die dezelfde rest $R(x)$ geven. Als een verminking optreedt, is er een kans dat zo'n verminkt bericht niet wordt gedetecteerd als zij dezelfde $R(x)$ heeft als het originele bericht. Deze kans hierop is afhankelijk van de Minimum Hamming Distance MHD in de groep berichten met dezelfde $R(x)$. Als een van de generatorpolynomen uit tabel 3.1 wordt genomen, dan worden alle 1-, 2- en oneven-aantal (dus ook 3) bit fouten gedetecteerd. De Minimum Hamming Distance is dan 4. Bij een optimale codering, zal men de Minimum Hamming Distance zelfs groter dan 4 kunnen maken.

Stel dat de kans op verminking van een willekeurig bit p is, dan is de kans dat een verminking van e bit in een codewoord $T(x)$ met de lengte $m+r$ bit niet herkend wordt, gelijk aan de kans dat het aantal foute bits $\underline{e}$ groter of gelijk is dan MHD:

$$P(\underline{e} \geq MHD) = \sum_{i \geq MHD} \binom{r+m}{i} p^i \cdot (1-p)^{r+m-i} \approx \binom{r+m}{MHD} p^{MHD} \cdot (1-p)^{r+m-MHD}$$

Hieruit blijkt dat de waarde van *m*, *r* en *MHD* bepalend zijn voor de kans op een *detectiefout*.

Hoewel het theoretisch mogelijk is om met een CRC met een Minimum Hamming Distance $MHD = \geq 3$ correctie te plegen op 1-bit fouten, wordt dat alleen gedaan als het bericht niet te lang is en het foutcorrectieproces voldoende tijd beschikbaar heeft. Meestal wordt CRC gecombineerd met andere methoden voor foutcorrectie, zoals *hertransmissie*. Na het detecteren van de fout met een CRC, wordt het bericht opnieuw verzonden.

3.2.4 Hammingcode

De eerste-orde Hammingcode is een methode die in staat is één-bit fouten te herkennen en te corrigeren. Zij is gebaseerd op meerdere pariteitcontroles over een aantal databits. Bij een woord van L bit zijn daarvoor minimaal $R = \lceil \mathrm{ld}\,(L+1) \rceil$ pariteitsbits nodig. De pariteitsbits $p_1, p_2, \ldots, p_R$ worden geplaatst op de posities $i = 2^n \quad n = 1 \ldots \lceil \mathrm{ld}\,(L+1) \rceil$ tussen de databits $d_i \quad i = 1 \ldots K$:

p_0	p_1	d_1	p_2	d_2	d_3	d_4	p_3	d_5	d_6	d_7	d_8	d_9	d_{10}
p_0		d_1		d_2		d_4		d_5		d_7		d_9	
	p_1	d_1			d_3	d_4			d_6	d_7			d_{10}
			p_2	d_2	d_3	d_4					d_8	d_9	d_{10}
							p_3	d_5	d_6	d_7	d_8	d_9	d_{10}

Het bronwoord 010011 wordt gecodeerd als 0101100011. Bij het decoderen wordt de pariteit gecontroleerd: $p_0 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_9$ en $p_1 = d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{10}$ tot en met p_4. Bij een 1-bit fout, (bijvoorbeeld 0101000011, een fout bit in de 5-de positie), volgt automatisch de positie van het foute bit ($p_0 p_1 p_2 p_3 = 0101_2 = 5_{10}$). Positie $p_0 p_1 p_2 p_3 = 0000_2$ is gereserveerd voor de foutloze toestand.

3.3 De continue kanaalcapaciteit

In deze paragraaf worden enkele begrippen besproken die eigenlijk thuis horen in de *signaalverwerkingstheorie*. Het begrip kanaalcapaciteit C uit de informatietheorie wordt gekoppeld aan de *fysische kanaalcapaciteit* of *continue kanaalcapaciteit* C_f uit de signaalverwerkingstheorie. De begrippen informatieoverdracht of onderlinge entropie $H(X;Y)$ uit de informatietheorie zijn verwant aan het begrip *transmissiesnelheid, informatiesnelheid* of *bitrate* R_f uit de signaalverwerking.

Als er geen sprake is van storingen dan is de fysische kanaalcapaciteit een maat voor de snelheid waarmee informatie wordt overgedragen. Uit de fysische signaalverwerking is bekend dat een storingsvrij kanaal met een bandbreedte W in eenheden Herz, een maximale overdracht ook wel *baud-rate* genoemd, heeft van $2 \cdot W$ herkenbare vermogenselementen per seconde. De bandbreedte is het verschil tussen de hoogste en de laagste

significante frequentiecomponent in het signaal. De eenheid van de transmissiesnelheid van de vermogenselementen wordt gegeven in de eenheid *baud*. Bij een gegeven codering en kanaalgedrag is de informatiesnelheid R_f:

$$R_f = 2 \cdot W \cdot H(X;Y) \qquad \text{[bit/sec]} \tag{3.7}$$

Voorbeeld 3.9

> Een laagfrequente telefoonverbinding heeft een bandbreedte van 3100 Hz. De telefoon verbinding gedraagt zich als een BSC met een foutkans $d = 0,1$ voor een digitale verbinding. De bron genereert een binaire code met de kansverdeling $p_0 = 0,6$ $p_1 = 0,4$. Uit de formules 3.2 en 3.3 volgt dat de onderlinge entropie $H(X;Y) = H(Y) - H(Y|X) = 0,981 - 0,469 = 0,512$ bit. De informatiesnelheid R_f is $2 \cdot 3100 \cdot 0,512 = 3174$ bit/sec.

Uit dit voorbeeld blijkt dat de informatiesnelheid hoger kan zijn dan de bandbreedte W. De uiterste grens voor de informatiesnelheid is de fysische kanaalcapaciteit C_f. Boven deze grens treedt *distorsie* of onherstelbare verminking op.

$$R_f \leq C_f \tag{3.8}$$

Om een indruk te krijgen van de fysische kanaalcapaciteit, moeten wij gebruik maken van fysische kenmerken van signalen zoals het vermogen van een signaal. Het vermogen van signalen wordt gegeven in Watt. Als het signaalvermogen in n^2 herkenbare vermogenspakketjes verdeeld kan worden, dan zijn $\sqrt{n^2} = n$ signaalniveaus herkenbaar en daarmee maximaal $\mathrm{ld}(n)$ bit informatie mogelijk. De fysische kanaalcapaciteit C_f voor een storingsvrij kanaal met een uniform verdeelde broncode wordt als volgt gedefinieerd:

$$C_f = 2 \cdot W \cdot \mathrm{ld}(n) = W \cdot \mathrm{ld}(n^2) \quad \text{[bit/sec]} \tag{3.9}$$

De eenheid van de fysische kanaalcapaciteit wordt gegeven in bits per seconde, dezelfde eenheid als de informatiesnelheid. Met deze definitie voor de fysische kanaalcapaciteit van een storingsvrij kanaal wordt dus geen rekening gehouden met storingen. Fysische storingsbronnen worden ook wel *ruisbronnen* genoemd. Ruisvermogen kan de oorzaak zijn dat individuele vermogenspakketjes niet herkenbaar meer zijn. Claude E. Shannon heeft aangetoond dat het aantal herkenbare vermogenspakketjes in een signaal afhankelijk is van de *signaal-ruis vermogensverhouding* P_S/P_N, waarbij P_S voor het vermogen van het signaal en P_N voor het vermogen van de ruis staat. Intuïtief zou men dit kunnen verklaren door aan te nemen dat de storingsbron en de signaalbron onafhankelijk van elkaar zijn. In dat geval mogen de vermogens (de varianties van de signalen) bij elkaar opgeteld worden. Het volledige signaal inclusief de ruis heeft nu een vermogen van $P_S + P_N$. Dit betekent dat het totale vermogen in $(P_S + P_N)/P_N$ herkenbare vermogenspakketjes verdeeld kan worden. Deze vermogenspakketjes staan voor $\sqrt{(P_S + P_N)/P_N}$ herkenbare

signaalniveaus en maken maximaal ld($\sqrt{1+P_S/P_N}$) bit informatie mogelijk. De fysische kanaalcapaciteit C_f voor een kanaal met storingsbron wordt als volgt gedefinieerd:

$$C_f = 2 \cdot W \cdot \mathrm{ld}(\sqrt{1+\tfrac{P_S}{P_N}}) = W \cdot \mathrm{ld}(1+\tfrac{P_S}{P_N}) \quad \text{[bit/sec]} \tag{3.10}$$

Voorbeeld 3.10

Bij een telefoonverbinding met een bandbreedte van $W_1 = 3100$ Hz wordt de signaal-ruisverhouding opgevoerd van 33 naar 54 dB. Als de fysische kanaalcapaciteit gelijk blijft, dan wordt de nieuwe bandbreedte W_2:

$$W_1 \cdot \mathrm{ld}(1+\tfrac{P_{S1}}{P_{N1}}) = W_2 \cdot \mathrm{ld}(1+\tfrac{P_{S2}}{P_{N2}})$$

Oplossen van W_2:

$$W_2 = \frac{3100 \cdot \mathrm{ld}(1+10^{\frac{33}{10}})}{\mathrm{ld}(1+10^{\frac{54}{10}})} = 1895 \quad \text{[Hz]}$$

Uit formule 3.10 blijkt dat als het signaalvermogen verhoogd wordt, bij gelijkblijvende fysische kanaalcapaciteit, de bandbreedte verlaagd kan worden. Men zou dit effect kunnen gebruiken om de fysische bandbreedt zeer nauw te maken of de kanaalcapaciteit te verhogen. Het is echter ondoenlijk om met versterkers het signaal/ruis vermogen onbeperkt op te voeren. Versterking van het signaal geeft ook versterking van de ruis.

Voorbeeld 3.11

Een telefoonverbinding met een signaal-ruisverhouding van 33 dB (decibel)[1] en een bandbreedte van 3100 Hz, heeft een fysische kanaalcapaciteit van:

$$C_f = W \cdot \mathrm{ld}(1+\tfrac{P_S}{P_N}) = 3100 \cdot \mathrm{ld}(1+10^{\frac{33}{10}}) = 3100 \cdot \mathrm{ld}(1996) = 34000 \quad \text{[bit/sec]}$$

Hieruit blijkt dat bij een normale telefoonverbinding een maximale informatieoverdracht van $5,48$ bit mogelijk is:

$$H(X;Y) = \frac{C_f}{2 \cdot W} = \frac{34000}{6200} = 5,48 \quad \text{[bit]}$$

Hoe wij aan de maximale onderlinge entropie kunnen komen, wordt niet meer in deze lessen informatietheorie behandeld. Moderne modems, die een diversiteit van coderings-compressie- en modulatietechnieken gebruiken, zijn in staat bij dit soort verbindingen transmissiesnelheden van 34 Kbit/sec en hoger te halen.

[1] De eenheid decibel wordt berekend als een logaritme met het grondtal 10 van de signaal-ruis verhouding: $10 \cdot \log_{10} \frac{P_S}{P_N}$

3.4 Opgaven

1. Een BSC heeft een *Binary Error Rate* van $0,3$. Wat is de discrete kanaalcapaciteit van dit kanaal?

2. Noem 3 decodeerprincipes en hun eigenschappen.

3. De *Soundex codering* is een codering die bronwoorden uit een West-Europesche spreektaal vertaalt in codewoorden van één letter gevolgd door drie cijfers. Het voordeel van deze codering is dat de woorden die veel op elkaar lijken qua uitspraak, dezelfde code krijgen. Soundex wordt veel toegepast in spellingscontrole in woordprocessoren, reisplanners en reserveringssystemen etc. Bijvoorbeeld, de namen *'brok', 'brock', 'broek'* geven dezelfde code *B*620. Daarentegen geven de namen *'jansen', 'janssen', 'jansens'* de code *J*525.

   Het Soundex-algoritme werkt als volgt:

   (a) De eerste letter van het bronwoord wordt de beginletter in het Soundex codewoord;

   (b) De volgende 3 cijfers komen uit de volgende tabel. Zij worden in volgorde opgebouwd in volgorde van de letters in het bronwoord. Als twee letters naast elkaar gelijk zijn, dan krijgen zij samen maar één cijfer. Als het bronwoord te kort is voor een volledig Soundex codewoord, wordt het Soundex codewoord aangevuld met nullen. Te lange codewoorden worden afgebroken na drie cijfers.

code	letter	uitspraakorgaan
1	b p f v	lippen
2	c s k g j q x z	keel
3	d t	tanden
4	l	tong voor
5	m n	neus
6	r	tong achter
geen	a e h i o u y w	

   Deze tabel geeft de cijfercodering van de letters aan. De meeste medeklinkers zijn volgens uitspraak gegroepeerd. De klinkers en de zachte medeklinkers krijgen geen cijfercode. Hoewel het Soundex-algoritme goed werkt voor West-Europese talen, is het niet voor andere spreektalen geschikt.

   (a) Welke aspecten maken Soundex codering geschikt voor West-Europese talen?

   (b) Een ander manier om alternatieve woorden te vinden is de '*Edit Distance*':

   - Spatie tussenvoegen;
   - Twee buurletters verwisselen;

- Een letter vervangen door een andere;
- Het verwijderen van een letter;
- Een letter toevoegen.

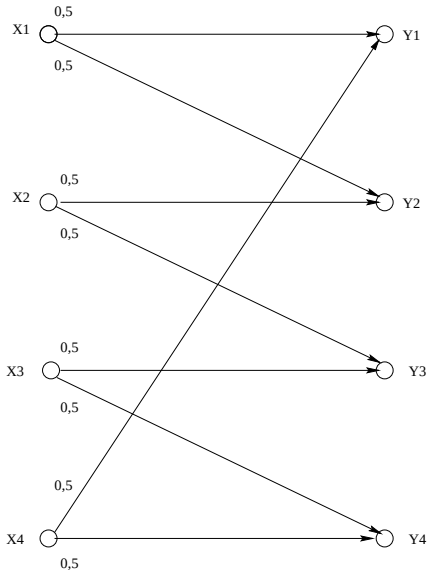
Wat is het belangrijkste verschil tussen de 'Edit Distance' en Soundex?

4. Gegeven een taal met 8 bronwoorden van 3 bit.

   (a) Hoeveel redundantie moeten de codewoorden bevatten om een taal met 8 bronwoorden van ieder 3 bit intolerant voor 2 bit fouten te maken?

   (b) Welk CRC-polynoom zou in aanmerking komen om de bronwoorden te beschermen tegen 1 en 2 bit fouten?

5. Een geheugenloos kanaal tussen X en Y heeft de volgende eigenschappen $P(Y = y|X = x) = 0,5$:



Figuur 3.5: Een kanaal met eenzijdige overspraak

   (a) Bepaal de discrete kanaalcapaciteit van het kanaal in figuur 3.5.

   (b) Ontwerp een transmissiecode waarmee via dit kanaal foutloze transmissie plaats kan vinden.

6. Een transmissiekanaal van 44000 [bit/sec] heeft een *Binary Error Rate* $d = 10^{-6}$, de kans dat een willekeurig bit omflipt. Er worden continue berichten van $m = 512$ bits verzonden met een CRC-16 ($r = 16$) met een Minimum Hamming Distance $MHD = 3$.

(a) Wat is de tijd waarin gemiddeld één detectiefout optreedt als de Minimum Hamming Distance $MHD = 3$?

(b) Wat is de tijd waarin gemiddeld één detectiefout optreedt als de Minimum Hamming Distance $MHD = 4$?

7. Wat is de kleinste CRC polynoom om 1-bit fouten te detecteren?

8. Hoeveel redundantie is er minimaal nodig om een bericht met $L = 8 \cdot 10^6$ te corrigeren voor 1-bit fouten:

   (a) Met horizontale en verticale pariteit?

   (b) Met CRC-4?

   (c) Met Hammingcode?

Hoofdstuk 4

Cryptografie

Al heel vroeg in de geschreven geschiedenis maakte men gebruik van geheimschrift of *cryptografie*. De Romeinse veldheer Julius Ceasar gebruikte bij de communicatie met zijn officieren geheimschrift. Daarmee voorkwam hij dat gevoelige informatie over zijn activiteiten en motieven bekend zou worden bij zijn vijanden. Dit waren niet zozeer de analfabetische Keltische- en Germaanse stammen, maar eigen manschappen die samenzwoeren met politieke concurrenten. Daarnaast verhinderde het geheimschrift dat iemand die zich voordeed als gevolmachtigde, valse boodschappen kon geven. In het krakersjargon noemt men dat *spoofing*. Bovendien was het moeilijk een gecodeerde boodschap te wijzigen zonder dat er getwijfeld werd over de authenticiteit. In het krakersjargon wordt het ongeoorloofd wijzigen van berichten of bestanden *tampering* genoemd.

De lettersymbolen in de *plain-texten* werden door Ceasar systematisch gesubstitueerd door andere lettersymbolen tot *cipher-texten*. Als een van de letters werd gevonden, dan waren de andere letters ook bekend. Omdat in de tijd van Ceasar toch al weinig mensen konden lezen, gaf deze *Ceasarcodering* een redelijke beveiliging. Al heel snel kwamen krakers er achter dat de letterfrequentie ongevoelig is voor deze substitutiemethode. Deze kennis leidde tot de ontwikkeling van sterkere coderingstechnieken zoals de transpositiecoderingen – het permuteren van de letters in het bericht

Tijdens de ontwikkeling van de cryptografie bleek al snel dat een sterke beveiliging tegen krakers meer codeerinspanning ging kosten dan een zwakke beveiliging. Tijdens de tweede wereldoorlog ontwikkelde men daarom cryptografische schrijfmachines zoals de ‘Enigma’ en later de ‘Hagelin-machine’. Dit soort machines waren ontworpen als typemachines met roterende codeschijven. Deze machines en de codeschijven kan men beschouwen als een cryptografisch algoritme. De sleutel werd gegeven door de codeschijven in te stellen in een bepaalde stand. Eenmaal ingesteld kon men het bericht door het in te typen snel coderen. De decryptie geschiedde op dezelfde wijze. Al spoedig bleken deze cryptografische machines minder sterk dan gedacht, de krakers (in dit geval de Poolse-, de Britse en de Amerikaanse geheime dienst) ging eveneens gebruik maken van machinale hulpmiddelen. Er ontstond een wedloop tussen de cipher-systemen en het ontcijferend vermogen van machines.

Tegenwoordig heeft men de beschikking over verschillende cipher-systemen die geschikt zijn voor machinale verwerking. Dit soort cipher-systemen bestaan uit twee deelsystemen, het *encryptiesysteem* en het *decryptiesysteem*, met een duidelijk en bewust gecreëerd onderscheid tussen het *algoritme* en de *sleutel*. Het encryptiesysteem vertaalt een plain-text naar een cipher-text aan de hand van een *encryptiesleutel*. Het decryptiesysteem vertaalt een cipher-text weer terug naar een plain-text met behulp van een *decryptiesleutel*. In een *symmetrisch cipher-systeem* zijn de encryptie- en decryptiesleutel aan elkaar gelijk. In een *asymmetrische cipher-systeem* zijn de encryptie- en decryptiesleutel verschillend. Zowel symmetrische- als asymmetrische cipher-systemen worden naast elkaar gebruikt omdat zij elkaars zwakheden compenseren.

In het algemeen is men ervan overtuigd dat, in tegenstelling tot de sleutels, een cryptografisch algoritme vrijelijk bestudeerd moet kunnen worden. Met andere woorden: "*Als de sleutel maar geheim is, mag het werkingsprincipe van het slot mag bekend zijn.*" Door de sleutels geheim te houden, zullen de gebruikers hun eigen beveiliging niet prijsgeven. Bovendien wint het cipher-systeem aan sterkte omdat iedereen de fouten en zwakheden van het algoritme kan signaleren en verbeteren.

De alternatieve strategie, "*Security by obscurity*", wordt tot schade van de klanten het meest beleden door leveranciers die hun kennis en zwakheden verborgen willen houden. Zij doen dit niet alleen voor de potentiële krakers maar uit concurrentieoverwegingen. Bovendien zijn zij bang voor reputatieverlies. Veiligheidsorganisaties zijn om andere redenen terughoudend over cryptografische kennis. Hun belang is niet gelegen in de verbetering van open cipher-systemen die misbruikt kunnen worden door criminelen. In het belang van de nationale veiligheid zullen zij evenmin laten blijken hoe groot hun cryptografische kennis is. Soms is deze veiligheidsbehoefte zo sterk dat voor cryptografische systemen dezelfde regels gelden die normaliter gelden voor wapensystemen. Anderzijds is een verbod op betrouwbare cryptografie moeilijk te verdedigen. "*Men verbiedt toch ook geen slot op een deur omdat de politie er niet door kan?*". Deze vraag wordt terecht gesteld omdat criminelen met een verbod op cryptografie beter af zijn dan niet-criminelen. Criminele activiteiten die geen aandacht mogen trekken, maken bovendien toch al gebruik van *steganografie*, het verbergen van boodschappen in- en tussen- andere berichten.

De cryptografie wordt niet alleen ten bate van de nationale veiligheid toegepast. Bij de bescherming van het elektronische gegevens-, handels en betalingsverkeer, de behoefte aan privacy en de bescherming van kennis, maakt men tegenwoordig gebruik van cryptografie. Gebruikers van communicatiemiddelen moeten kunnen vertrouwen op de vertrouwelijkheid, integriteit en beschikbaarheid van gegevens. Bovendien is het voor het handelsverkeer noodzakelijk maatregelen te nemen tegen spoofing en tampering door te bewijzen dat de verzender van gegevens degene is voor wie die zich uitgeeft ('prove of origin') en dat een derde partij verstuurde gegevens ontvangen heeft ('prove of receipt'). De ontwikkeling van *Internet* en *E-commerce* is in sterke mate afhankelijk van de vrije verkrijgbaarheid van cryptografie. Het zal duidelijk zijn dat handels- en veiligheidsbelangen soms tegenstrijdige eisen stellen.

4.1 Aanvallen op de cryptografische bescherming

Een aanval op een cipher-systeem wordt uitgevoerd door een *kraker* die een gecodeerd bericht wil ontcijferen of aanpassen. Meestal richt de aanval zich rechtstreeks op het gecodeerde bericht, soms richt de aanval zich op de sleutel(s). In dit laatste geval kan de kraker meer berichten ontcijferen of versturen. Het met behulp van 'brute force' kraken van een code bestaat uit het stomweg uitproberen van sleutels totdat er een is waarmee de gegeven cipher-text gekraakt kan worden. De inspanning die men moet plegen om zo'n aanval met succes uit te voeren, is evenredig met de entropie van de sleutel $H(K)$. Entropie is immers het aantal ja-nee antwoorden waarmee een bericht met zekerheid vast te stellen is. 'Brute force' aanvallen leiden alleen bij sleutels met beperkte entropie tot succes. De ondergrens van de 'brute force' tijd wordt bepaald door de meest fantastische computer die op deze aarde mogelijk is. Zij gebruikt alle beschikbare energie en ruimte op aarde en kan maximaal 10^{38} instructies per seconde uitvoeren zonder de bekende fysische wetten te trotseren. De volgende tabel geeft een indruk van de prestaties van dit rekenmonster:

bit	tijd
75	$3,8 \cdot 10^{-16}$ seconden
100	$1,3 \cdot 10^{-8}$ seconden
125	$0,4$ seconden
150	165 dagen
175	$15 \cdot 10^{6}$ jaren
200	$5 \cdot 10^{14}$ jaren

Omdat alle hedendaagse computers bij elkaar minder krachtig zijn dan dit futuristische rekenmonster, kunnen zij zelfs gemeenschappelijk een 'brute force' aanval niet sneller uitvoeren. De grenzen van de computerverwerking beperken de mogelijkheden van de kraker. Dit noemt men *computational security*. '*Brute force*' aanvallen op sleutels met een entropie van meer dan 75 bit zijn op dit moment niet zinvol. Echter, gebruikers van computersystemen kiezen vaak voor zwakke sleutels. In zulke gevallen tracht men het kraken te bemoeilijken door de beschikbare tijd voor de kraker te verlagen.

Voorbeeld 4.0

Een inlogprocedure op een computer wordt beperkt tot één poging per seconde met een maximum van drie mislukte pogingen. Dit voorkomt dat een kraker met een automatische sleutelgenerator in korte tijd veel sleutels kan proberen.

Voorbeeld 4.1

In een stroom van korte berichten wordt elk bericht apart gecodeerd met een unieke sleutel. Voordat een van deze sleutels gekraakt wordt, is hij al onbruikbaar.

Een kraker moet meestal naar andere middelen grijpen dan de 'brute force' aanpak om binnen de beschikbare tijd een sleutel te vinden. In het algemeen zal een kraker elke bit informatie kunnen gebruiken die kan leiden tot de ontcijfering van een code. Met Bayesiaanse kansberekening zijn effectieve aanvallen mogelijk. De belangrijkste tactieken van de kraker zijn de volgende aanvallen:

Cipher-text-only-attack: Hierbij heeft de kraker de beschikking over een cipher-text. Door wiskundige en statistische analyse zal de kraker een plain-text trachten te ontcijferen. Voor een kraker is dit de moeilijkste aanval. Bij een zogenaamd '*one-time-pad*' code is de kans op ontcijfering nihil. Daarentegen is bij een Ceasarcode van voldoende lengte een 'cipher text only attack' niet ondoenlijk;

Known-plain-text-attack: De kraker kent een gedeelte van een bericht en de gecodeerde versie daarvan. Aan de hand van deze gedeeltelijke informatie zal de kraker de rest van een plain-text trachten te ontcijferen. Meestal weet een kraker dat bepaalde onderdelen van plain-texten identiek zijn. Het is niet onwaarschijnlijk dat een kraker vanuit andere informatiebronnen de beschikking heeft gekregen over de namen van de afzender en de ontvanger en andere voor de handliggende tekstonderdelen. De kansen op ontcijfering worden groter naarmate er meer informatie gegeven is en naarmate de kraker meer cipher-texten ter beschikking staan;

Chosen-plain-text-attack: De kraker heeft een zelf gekozen bericht en de gecodeerde versie daarvan. Dit is een gevaarlijke aanval, zeker als de kraker in staat is zelf gekozen teksten te coderen. De 'chosen plain-text attack' wordt uitgevoerd door een kraker die op een of ander manier de toegang heeft gekregen tot het encryptieproces zonder de sleutel te kennen. Meestal lukt dit door medewerking aan- of uitlokking van encryptieopdrachten.

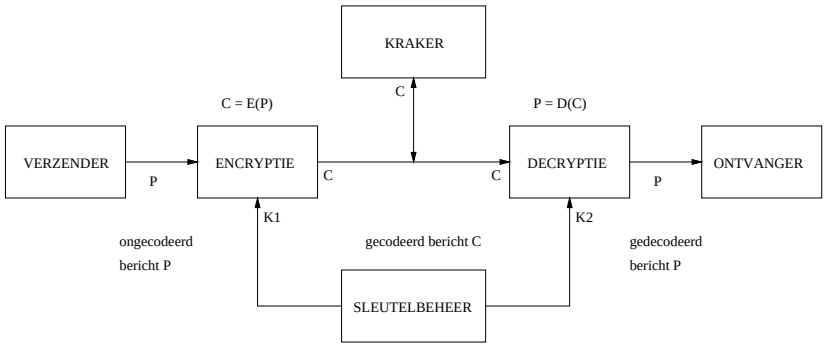
4.2 Formele cryptografie

Met behulp van de informatietheorie kan de externe sterkte en zwakte van een cipher-systeem geanalyseerd worden. Bij de analyse van de interne sterkte en zwakte van een cipher-systeem moet men gebruik maken van specialistische wiskunde zoals groepenalgebra, combinatoriek en getaltheorie. Wij beperken ons daarom tot algemene beschrijving van de cryptografie.

In formele termen bestaat een cipher-systeem uit een encryptie-algoritme E, dat een plain-text P vertaalt naar een cipher-text C:

$$C = E_K(P)$$

Bovendien bestaat het uit een decryptie-algoritme D dat een cipher-text C terugvertaalt naar een plain-text P:



Figuur 4.1: Schema van communicatie met cryptografisch codering

$$P = D_K(C)$$

In de formele cryptografie worden alle mogelijke plain-texten weergegeven als de variabele P die $|P|$ uitkomsten heeft. De entropie van het bericht $H(P)$ wordt bepaald door de kansen op individuele berichten P_i. De entropie voor een bericht is maximaal als de individuele berichten onafhankelijk en uniform verdeeld zijn.

$$H(P) = -\sum_{i=1}^{|P|} p(P_i) \cdot \mathrm{ld}\,(p(P_i))$$

Alle mogelijke cipher-texten worden weergegeven als de variabele C met $|C|$ uitkomsten. Normaal zijn het aantal mogelijke plain-texten en cipher-texten gelijk aan elkaar $|P| = |C|$. Voor de entropie van een cipher-text $H(C)$ geldt de volgende uitdrukking:

$$H(C) = -\sum_{i=1}^{|C|} p(C_i) \cdot \mathrm{ld}\,(p(C_i))$$

De variabele K met $|K|$ uitkomsten staat voor alle mogelijke sleutels. De waarde $|K|$ is het maximale aantal sleutels. Dit komt overeen met het maximale aantal pogingen die een kraker met een ‘brute force’ methode moet doen om de sleutel te raden. Meer nog dan de maximale waarde $|K|$ en de daaruit volgende maximale entropie $H_{max}(K) = \mathrm{ld}\,(|K|)$, is de werkelijke entropie $H(K)$ het echte kwaliteitscriterium van de sleutelverdeling. De ‘brute force’ inspanning komt, zoals eerder gezegd, overeen met de entropie $H(K)$, de hoeveelheid gemiddelde informatie. Een kraker zou maximaal $2^{H(k)}$ keer moeten gokken om de sleutel te vinden.

$$H(K) = -\sum_{i=1}^{|K|} p(K_i) \cdot \mathrm{ld}\,(p(K_i))$$

Indien de sleutels willekeurig gekozen worden uit de verzameling strings met een lengte van 8 willekeurig gekozen tekens (26 letters, 26 hoofdletters, 10 cijfers en zo'n 30 leestekens) dan zou de maximale entropie $H_{max}(K)$ van deze sleutelkeuze $8 \cdot \mathrm{ld}\,(92) = 52,19$ bit zijn. Bij deze uniforme sleutelverdeling zou een kraker maximaal $2^{52,19} = 5 \cdot 10^{15}$ ja-nee pogingen moeten uitvoeren om de sleutel te raden.

Als echter bekend is dat een sleutel één van de $65000 \approx 2^{16}$ zelfstandige naamwoorden in een Nederlands woordenboek is, dan is deze sleutel met maximaal 65000 pogingen redelijk snel te vinden. Gemiddeld zal het binnen de helft van de tijd wel lukken. De entropie van deze sleutelverdeling is slechts 16 bit. Hoewel dit voorbeeld duidelijk een slechte sleutelverdeling heeft, worden in de praktijk meestal bestaande woorden of namen gekozen als sleutel.

Voorbeeld 4.2

Een Ceasarcodering substitueert de lettersymbolen met andere lettersymbolen. Deze substitutie is een zeer eenvoudige verschuiving van letters in het alfabet, bijvoorbeeld de 'a' wordt een 'o', de 'b' wordt een 'p' etc. In dit voorbeeld worden de letters in dit 13 plaatsen in het alfabet verschoven. De substitutie van de i^{de} letter door de j^{de} letter wordt formeel beschreven met de formule: $(i = j + t) \bmod 26$. De sleutel t, de lengte van de verschuiving in het alfabet, kan 26 mogelijke waarden aannemen. Daaruit volgt dat de entropie van de sleutel $H(K)$ maar $\mathrm{ld}\,(26) = 4,7$ bit is.

Voorbeeld 4.3

Een '*monoalfabetische substituering*' maakt geen gebruik van een verschuiving, zij permuteert de lettersymbolen in het alfabet. Er zijn 26! permutaties mogelijk als sleutel. De entropie van de sleutel $H(K)$ is $\mathrm{ld}\,(26!) = 88,4$ bit. Het lijkt of de sleutel een hoge entropie heeft. Maar bij een gecodeerde natuurlijke tekst kan een kraker, met gebruikmaking van nulde- en hogere orde letterfrequenties, de sleutel redelijk snel vinden.

Voorbeeld 4.4

Een '*polyalfabetische substituering*' maakt gebruik van een of meer monoalfabetische substituties. De entropie van de sleutel is afhankelijk van de polyalfabetische sleutel van letters en het aantal monoalfabetische substituties. Ook hier geldt dat bij natuurlijke teksten een kraker, met gebruikmaking van taaleigenschappen die ongevoelig zijn voor permutaties en verschuivingen, de sleutel kan vinden. Een eenvoudig voorbeeld van een polyalfabetische codering is de *Vignerecodering*. De Vignerecodering vertaalt een letter m in een letter c_n op basis van zijn positie n in de tekst, de sleutel K (een string met de lengte k), met de formule: $(c_n = m_n + K(n(n) \bmod k)) \bmod 26$. Voor het gemak wordt aangenomen dat er maar 26 letters zijn, genummerd van $0 \ldots 25$. De positie n van de letter is genummerd van $0, 1, 2, \ldots$.

Het gebruiken van informatie zoals de letterfrequentie in een cipher-text, de zogenaamde 'cipher-text-only-attack' op een sleutel, wordt moeilijker als informatie over de sleutel K niet zondermeer uit een cipher-text C te destilleren is. Daarom moet de *key equivocation* $H(K|C)$ een hoge entropie bevatten:

$$H(K|C) = -\sum_{h=1}^{|K|}\sum_{i=1}^{|C|} p(K_h,C_i)\cdot \mathrm{ld}\,(p(K_h|C_i))$$

Gemakkelijker dan een 'cipher-text-only-attack' op de sleutel is een 'cipher-text-only-attack' op een plain-text. Een plain-text hoeft geen gebruik te maken van alle lettersymbolen. Het is voldoende om dat gedeelte van de sleutel te vinden waarmee een plain-text vertaald kan worden. De inspanning van deze aanval is evenredig met de *plain-text-equivocation* $H(P|C)$:

$$H(P|C) = -\sum_{j=1}^{|P|}\sum_{i=1}^{|C|} p(P_j,C_i)\cdot \mathrm{ld}\,(p(P_j|C_i))$$

Voor de plain-text-equivocation geldt dat zij nooit hoger is dan de key equivocation $H(P|C) \leq H(K|C)$. Dit geldt ook voor de inspanning van de 'known-plain-text-attack', de entropie van de '*key-appearance-equivocation*' $H(K|C,P)$:

$$H(K|C,P) = -\sum_{h=1}^{|K|}\sum_{i=1}^{|C|}\sum_{j=1}^{|P|} p(K_h,C_i,P_j)\cdot \mathrm{ld}\,(p(K_h|C_i,P_j))$$

Omdat de kraker informatie uit een cipher-text en een overeenkomstige plain-text kan gebruiken zal de entropie $H(K|C,P)$ niet groter zijn dan $H(K|C)$. Indien de kraker de beschikking heeft over een sleutel K, kan een cipher-text C zonder problemen gedecodeerd worden tot een plain-text P. Hieruit volgt dat P volledig afhankelijk is van C en K, (zie formule 1.15):

$$H(P|C,K) = 0$$

Tussen de verschillende equivocaties bestaan de volgende relaties:

$$\overbrace{H(P|C,K)}^{0} + H(C,K) = H(P,C,K) = H(K|P,C) + H(P,C)$$

Daaruit volgt:

$$H(C,K) = H(K|P,C) + H(P,C)$$

Na de volgende afleiding:

$$H(K|P,C) = H(K|C) + H(C) - H(P|C) - H(C) = H(K|C) - H(P|C)$$

vinden wij de relatie tussen de equivocaties:

$$H(K|P,C) = H(K|C) - H(P|C) \tag{4.1}$$

Om te voorkomen dat een 'known-plain-text-attack' te gemakkelijk is, zal men $H(K|P,C)$ zo groot mogelijk willen maken. Men mag echter niet het risico op een geslaagde 'cipher-text-only-attack' verhogen door de sleutelequivocatie $H(P|C)$ te verlagen. Er blijft geen andere mogelijkheid over dan de entropie $H(K|C)$ zo groot mogelijk te maken. Uit een cipher-text mag geen informatie over de sleutel te vinden zijn. Toch blijken aanvallen mogelijk die gebaseerd zijn op de verlaging van $H(K|C)$ door zoveel mogelijk ciphertexten te gebruiken.

$$H(P;C) = H(P) - H(P|C) = H(C) - H(C|P) \tag{4.2}$$

Ook de onderlinge informatie $H(P;C)$ zal men zo klein mogelijk willen maken om te voorkomen dat zonder sleutel uit een cipher-text informatie over een plain-text te verkrijgen is en vice versa. Dit kan door de sleutel onafhankelijk te maken van het bericht en een plain-text niet afhankelijk te maken van een cipher-text. Een codering waarin een plain-text zelf gebruikt wordt voor versleuteling voldoet niet aan deze eis. Zo'n cipher-text kan gekraakt worden met autocorrelatietechnieken waardoor informatie over de sleutel K is te verkrijgen. Een sleutel K zou in principe maar één keer gebruikt mogen worden. Voor absoluut veilige cipher-systemen geldt:

$$H(P;C) = 0 \tag{4.3}$$

Uit de relaties:

$$H(P;C) = H(P) - H(P|C) \geq H(P) - H(K|P) \geq H(P) - H(K) \tag{4.4}$$

volgt dat voor een absoluut veilig cipher-systeem moet gelden:

$$H(K) \geq H(P) \tag{4.5}$$

De laatste formule bewijst dat een berichtenstroom waarin steeds dezelfde sleutel gebruikt wordt, in principe gemakkelijker gekraakt kan worden indien de som van de berichten groter is dan de sleutellengte. Volledige veiligheid kan pas gegarandeerd worden als de sleutel gelijk of langer is dan het bericht (of de berichtenstroom). Het '*one-time-pad*' (*OTP*) voldoet aan de eis, mits bij elke berichtenstroom een nieuwe sleutel gekozen wordt.

Wat is een ‘one-time-pad’ voor codering? Stel dat een plain-text P een lengte heeft van L symbolen uit een alfabet met α symbolen. Een sleutel bestaande uit L onafhankelijke symbolen, kan met een simpele rekenkundige operatie gebruikt worden om uit een plain-text P een cipher-text C te maken:

$$C_i = (P_i + K_i) \text{ modulo } \alpha$$

Door dit nogal evenvoudige algoritme is een ‘one-time-pad’ bijzonder efficiënt en wordt vaak met behulp van speciale hardware uitgevoerd.

Voorbeeld 4.5

> Het bericht $[045154]$ uit het alfabet $\{0,1,2,3,4,5\}$ met $\alpha = 6$ symbolen, wordt gecodeerd met de OTP-sleutel $[335124]$ tot de cipher-text $[314212]$.

Een OTP-sleutel moet een hogere entropie hebben dan het bericht. Alleen in dat geval is de onderlinge informatie $H(P;C) = 0$. Dit betekent dat een cipher-text:

```
qC(dF-aZc6v=#nEd*sp
```

met evenveel kans ontcijferd kan worden tot een van de volgende zinnen:

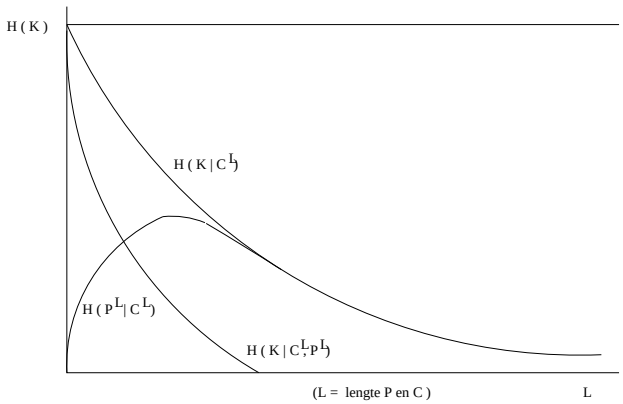
```
het antwoord is ja!
het antwoord is nee
vergeet het maar :)
```

Als een cipher-text met verschillende sleutels gedecodeerd kan worden tot verschillende plain-texten, noemen wij de codering een *perfecte codering*. De kraker kan nooit aan de hand van een plain-text bepalen of de juiste sleutel gevonden is. OTP is een perfecte codering.

Tot nu toe is er geen rekening gehouden met het aantal cipher-texten dat een kraker tot zijn beschikking heeft. Zoals eerder gezegd zal een kraker elke bit informatie kunnen gebruiken om een kansloze ‘brute force’ aanval te voorkomen. Met behulp van letterfrequenties, correlatietechnieken, formule van Bayes en dergelijke, neemt de kans op succes toe met het aantal bekende cipher-texten. Neem aan dat het aantal cipher-texten gelijk is aan L. De notatie C^L staat dan ook voor een samengesteld cipher-text met de lengte L.

De key equivocation $H(K|C^L)$ neemt af met L. Uiteindelijk zal voor een grote waarde van L de ‘*key equivocation*’ tot nul naderen. In figuur 4.2 wordt dit effect grafisch weergegeven.

Voor de ‘plain-text-equivocation’ $H(P^L|C^L)$ treedt een ander effect op. In eerste instantie, bij kleine berichtlengte L, is de entropie van een bericht $H(P^L)$ nog klein. Daaruit volgt



Figuur 4.2: Key-, plain-text-, en key-appearance equivocation als functie van L

dat $H(P^L|C^L)$ (zie formule 1.14) kleiner of gelijk is. De 'plain-text-equivocation' neemt toe met de lengte L totdat de informatie uit de cipher-texten C^L de overhand krijgt.

De 'key appearance equivocatie' $H(K|C^L,P^L)$ neemt sneller af dan de 'key equivocatie' $H(K|C^L)$ omdat de kraker niet alleen de beschikking heeft over cipher-texten maar ook over plain-texten. Deze informatie zal een kraker eerder tot een geslaagde aanval brengen. Uiteindelijk nadert bij toenemende lengte L de waarde $H(K|C^L)$ tot nul.

Dit leidt ons tot de vraag hoeveel cipher-texten zijn nodig voor een succesvolle aanval? Om deze vraag te beantwoorden moeten wij de relatie tussen L en $H(K)$ onderzoeken. Uit de volledige afhankelijkheid tussen $H(K,C^L)$ en $H(K,P^L)$ volgt dat:

$$H(K|C^L) = H(K,C^L) - H(C^L) = H(K,P^L) - H(C^L) = H(K) + H(P^L) - H(C^L) \quad (4.6)$$

Wij kunnen zondermeer aannemen dat de entropie van een samengesteld cipher-text van de lengte L, niet meer entropie bevat dan L symbolen uit een uniform verdeeld broncode alfabet $H(C^L) \leq L \cdot H_{max}(P)$. Bovendien heeft een nulde-orde plain-text van de lengte L evenveel entropie als $L \cdot H(P)$, dit betekent dat $H(P^L) = L \cdot H(P)$. Tenslotte is het niet mogelijk dat de entropie $H(C^L)$ negatief wordt. Uit formule 4.6 en:

$$0 \leq H(C^L) \leq L \cdot H_{max}(P) \quad \wedge \quad H(P^L) = L \cdot H(P)$$

en het feit dat bij voldoende grote L, de waarde $H(K|C^L)$ tot nul nadert:

$$0 \leq H(K) + L \cdot H_{max}(P) - L \cdot H(P)$$

kunnen wij concluderen dat:

$$L \geq \frac{H(K)}{H_{max}(P) - H(P)} \tag{4.7}$$

De *kritieke lengte* is de kleinste waarde van L die voldoet aan formule 4.7. Zij wordt de *Unicity Distance* UD genoemd:

$$UD = \frac{H(K)}{H_{max}(P) - H(P)} = \frac{H(K)}{R(P)} \tag{4.8}$$

Waarin de variabele $R(P)$ de absolute redundantie (zie formule 1.21) van de bron P is.

Voorbeeld 4.6

> Een natuurlijke tekst in het Nederlands heeft een relatieve nulde-orde redundantie van 50%. Een Nederlandse tekst wordt op karakterbasis monoalfabetisch versleuteld. De absolute nulde-orde redundantie is $R(P) = 0,5 \cdot \mathrm{ld}\,(26) = 2,35$ bit. Minimaal zijn $UD = H(K)/R(P) = 88,4/2,35 = 37,6$ karakters nodig voor een geslaagde aanval. Bij een hogere orde overvloedigheid van 80% wordt de kritieke lengte verlaagd tot $UD = H(K)/R(P) = 88,4/3,76 = 23,5$ karakters.

Indien een cipher-text boven de kritieke lengte komt, is er maar één sleutel te vinden waarmee een cipher-text zinvol gedecodeerd kan worden. Het is geen maat voor de 'brute-force' inspanning die daarvoor noodzakelijk is. Wij kunnen de kritieke lengte UD verhogen door $R(P)$ te verlagen met compressietechnieken.

Voorbeeld 4.7

> Een natuurlijke tekst in het Nederlands is gecomprimeerd met een *code-efficiëntie* van 99,8%, de relatieve redundantie is 0,2%. De gecomprimeerde Nederlandse tekst wordt op karakterbasis monoalfabetisch versleuteld. De absolute redundantie is $R(P) = 0,02 \cdot \mathrm{ld}\,(26) = 0,0094$ bit. Minimaal zijn $L \geq H(K)/R(P) = 88,4/0,0094 = 9403$ karakters nodig voor een geslaagde aanval.

Voorbeeld 4.8

> *DES* (*Data Encryption Standard*) codeert een bericht in blokken van 64 bit met dezelfde 56 bit sleutel. Als wij de maximale entropie van een natuurlijke tekst $H_{max}(P)$ voor het gemak gelijkstellen aan de entropie van een cipher-text $H(C)$, dan is de kritieke lengte 1,09 blok (70 bit). Ondanks alle geruststellende verklaringen van de NSA[1], werd DES in 1998 gekraakt met zeer goedkope apparatuur.

[1] De '*National Securety Agency*' is een overheidsbureau van de Verenigde Staten, dat belast is met de ontwikkeling en de controle op cryptografie. Dit bureau is medeverantwoordelijk voor de ontwikkeling en de popularisering van DES.

4.3 Interne eigenschappen van cipher-systemen

Gevaarlijke externe statistische aanvallen zijn meestal gebaseerd op informatie over letterfrequenties en veel voorkomende woorden en zinsdelen. Een cipher-systeem moet gebaseerd zijn op twee principes die beide noodzakelijk zijn voor de robuustheid.

Het eerste principe, de *diffusie,* is het vervlakken van de letterverdeling. Op deze manier krijgen de letterfrequenties in de cipher-text min of meer gelijke waarden. Diffusie kan men bereiken door een letter in een cipher-text afhankelijk te maken van zoveel mogelijk andere letters in de plain-text. Of met andere woorden, een letter in de plain-text bepaalt vele letters in de cipher-text. Diffusie is bijvoorbeeld mogelijk door de som te nemen van een rij letters:

$$c_n = \sum_{i=n}^{k+n} m_i \text{ modulo } 26$$

Ook hoger-orde frequentieverdelingen moeten door de diffusie vlakker worden. Omdat het niet erg efficiënt is alle letters te betrekken bij diffusie, wordt de plain-text in blokken verdeeld. Bij *blokvercijfering* worden eerst de letters per blok gepermuteerd, vervolgens wordt het blok met een vervlakkend algoritme bewerkt.

Indien een kraker toch in staat is letterfrequenties te achterhalen, moeten cipher-system steunen op een tweede principe, de *confusie*. Confusie probeert de afhankelijkheid tussen een cipher-text en een sleutel zo laag mogelijk te maken. Dit kan door ingewikkelde substituties toe te passen. Deze ingewikkeldheid wordt verkregen door meerdere-, onafhankelijke-, verschillende bewerkingen na elkaar uit te voeren. Vaak worden bepaalde gecombineerde bewerkingen een paar keer herhaald. Dit soort cipher-systemen worden *productciphers* genoemd.

Naast de statistische aanvallen op de externe en interne zwakheden van cipher-systemen zijn andere aanvallen op de sleutel mogelijk. In het bijzonder zijn de zwakheden van het sleutelbeheer en de authenticatieproblemen de een van de oorzaken dat *e-commerce* niet die vlucht genomen heeft die men er van verwacht had. Hoe zou je ooit via een onbetrouwbaar kanaal zoals Internet, vertrouwen kunnen krijgen in een onbekende partij?

4.4 Sleutelbeheer

Het *sleutelbeheer* is vaak de zwakke schakel van de beveiligingsketen. Krakers zullen hun aandacht eerder richten op beheer van de sleutels dan op het kraken van de sleutels en cipher-texten zelf. De aanval op het sleutelbeheer kan zich richten op het genereren, het gebruik, de opslag en de overdracht van sleutels.

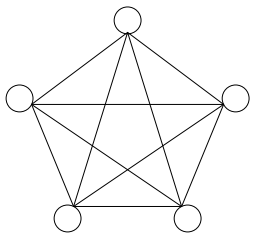
Sleutels worden vaak gegenereerd met een bepaalde methode. Personen die toegangscodes moeten onthouden kiezen vaak voor een naam en/of geboortedatum van een familielid

of huisdier. Krakers kennen deze voorkeur, zij hebben niet veel werk om dit soort sleutels te vinden. Een methode die sleutels genereert, mag in principe niet deterministisch zijn. Men krijgt met het werpen van dobbelstenen hogere entropische waarden dan met bijvoorbeeld een software randomgenerator, of een datum/tijd methode.

Voorbeeld 4.9

> Een *maximumlengtereeks-randomgenerator* kan een rij bits met een periodelengte van $2^{32}-1$ genereren. De genereerde bitreeks is volledig afhankelijk van de 32 bit startwaarde. Hoewel de gegeneerde reeks zeer lang kan zijn voordat zij zich herhaalt, is de entropie van deze reeks niet veel hoger dan de entropie van de startwaarde. De totale entropie van de rij van $4,3 \cdot 10^9$ bits is niet meer of minder dan de 32 bit entropie van de startwaarde. In veel programma's gebruikt men de standaardfunctie *random* om een pseudorandom rij getallen te berekenen. Het zal duidelijk zijn dat zo'n deterministisch bepaalde rij getallen een veel lagere entropie bezit dan de periodelengte van zo'n reeks suggereert.

Kwantumeffecten zijn theoretisch in staat tot het genereren van sleutels met hoge entropie. Tijdens het gebruik van een sleutel kan een kraker het invoeren observeren. De aanvallen op de opslag van sleutels kan men afslaan door deze sleutels op een fysiek (afgeschermd tegen alle mogelijke vormen van spionage) veilige plaats te bewaren en te gebruiken. Het bewaren van een sleutel op een computer die aangesloten is op een netwerk, is vragen om moeilijkheden. Bovendien moeten zo min mogelijk personen de sleutel kennen. Indien men het niet te nauw neemt met de beveiliging, zoals een briefje met een toegangscode laten slingeren, is de sleutel snel bekend bij onbevoegden. Anderzijds kunnen paranoïde personen heel ver gaan in hun streven naar optimale beveiliging. Zij overschrijven herhaaldelijk hun harde schijf met speciale wispatronen om te voorkomen dat een kraker achter hun geheime informatie kan komen.



Figuur 4.3: Maximaal $\binom{5}{2} = 10$ verbindingen tussen 5 partijen

Bij een beveiligd communicatiekanaal moeten bij een symmetrische cryptografische codering de zender en de ontvanger gebruik maken van dezelfde sleutel. Daarmee komen wij op de vraag hoe de sleutel van de een aan de ander wordt overgedragen. Dit zal in eerste instantie via een onbeveiligd communicatiekanaal moeten of via een alternatief beveiligd kanaal. Bovendien mag deze unieke sleutel alleen geldig zijn voor de beperkte tijd van de

communicatie. Ook mag deze sleutel niet meer gebruikt worden voor communicatie met andere partijen. Als er op de wereld 10^9 partijen zijn, dan zijn er in theorie $\binom{10^9}{2} \approx 5 \cdot 10^{17}$ verbindingen mogelijk, waarbij per verbinding een sleutel noodzakelijk is. In de praktijk van het normale handelsverkeer moet elke partij met *n* relaties rekening houden met de generatie, de opslag, het beheer en de overdracht van in de orde van $\binom{n}{2}$ sleutels. Dit zal in de praktijk te leiden omvangrijke kosten en risico's ten aanzien van het sleutelbeheer.

Symmetrisch cryptografische beveiligingen waarbij de sleutels eerst uitgewisseld moeten worden, hebben om deze reden een beperkte toepassing. Het sleuteluitwisselingsprobleem wordt grotendeels opgelost door het toepassen van publieke sleutels.

4.5 Publieke sleutelsystemen

Een van de eerste oplossingen voor het sleuteluitwisselingsprobleem werd geformuleerd door Whitfield Diffie en Martin Hellman. Dit systeem kan veilig een one-time-pad sleutel uitwisselen over een onveilig kanaal. Beide partijen spreken via het onveilige kanaal af welke gemeenschappelijke éénwegsfunctie (ondoenlijk inverteerbare functie) zij zullen gebruiken om hun gemeenschappelijke sleutel te genereren. Vervolgens zal elke partij zijn geheime sleutel verwerken met deze éénwegsfunctie en het resultaat via het onveilige kanaal naar de andere partij versturen, waaruit de andere partij met zijn geheime sleutel de gemeenschappelijke sleutel kan genereren. Het belangrijkste kenmerk van deze *Diffie-Hellman sleuteluitwisseling* is dat een veilige uitwisseling van sleutels via een onveilig kanaal mogelijk is. Het zwakke punt van deze uitwisseling is een derde partij die zich voordoet als een van de partijen ('de *man in het midden*'). Om de authenticiteit van een partij te bepalen, zijn meer maatregelen noodzakelijk. Oplossingen worden gezocht in het publieke sleutelsysteem en certificering.

Elke partij van een publiek sleutelsysteem heeft een eigen algemeen bekende sleutel. Maar hoe weet je zeker dat het de betreffende partij is? Zonder meer een publieke sleutel van Internet afhalen is gevaarlijk, er zijn personen die valse publieke sleutels kunnen verspreiden. Maar laten wij aannemen dat er een instantie is die garandeert dat een publieke sleutel aan een partij toebehoort. Dan kunnen wij zo'n publieke sleutel $K1$ gebruiken om aan zo'n partij een cipher-text te sturen. Bovendien verlaagt dit de complexiteit van het sleutelbeheer. Voor *n* partijen zijn er slechts *n* publieke sleutels nodig.

$$C = E_{K1}(P)$$

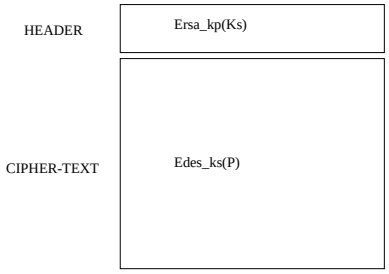
Hoewel de verzendende partij de beschikking heeft over een plain-text P en een cipher-text C gecodeerd met publieke sleutel $K1$, is een cipher-text alleen te decoderen door een geheime sleutel $K2$. Deze sleutel is alleen bekend aan de ontvangende partij. Een kraker die een cipher-text kent, is niet in staat het bericht te decoderen zonder de geheime sleutel $K2$.

$$P = D_{K2}(C)$$

Een publiek sleutelsysteem is te vergelijken met het verspreiden van kistjes met open hangsloten onder je relaties. Als zij je iets willen toezenden dan doen zij dat in zo'n kistje en sluiten het hangslot. Zij kunnen er dan niet meer bij. Alleen jij als eigenaar van de hangsloten kan ze weer open maken met de sleutel die je in bezit hebt gehouden.

In de praktijk zijn publieke sleutelsystemen nogal inefficiënt bij elektronische communicatie. Men past om daarom vaak een combinatie van publieke sleutels en een efficiënter symmetrisch systeem toe. Bij de communicatie via een *secure socket layer* (*SSL*) of een *secure shell* (*SSH*) gebruikt men een publiek sleutelsysteem om symmetrische 'one-time-pad' sleutels uit te wisselen. Met deze uitwisseling voorkomt men een belangrijk nadeel van het 'one-time-pad' systeem, de uitwisseling van sleutels. Bij vrij-verkrijgbare SSH en SSL wordt een betrouwbaar publiek sleutelsysteem zoals RSA gecombineerd met betrouwbare efficiënte symmetrische sleutelsystemen zoals (three-key *triple-DES*, *DES*, *RC4-128*, *Blowfish*, *Twofish*). Elk uur wordt de symmetrische sleutel vernietigd en een nieuwe symmetrische sleutel uitgewisseld.

Het systeem van publieke sleutels heeft naast een minder complex sleutelbeheer een tweede voordeel. Het is mogelijk een plain-text zo te coderen dat de ontvanger weet dat het van een bepaalde partij afkomstig is. Het bericht krijgt een authentieke status met een *elektronische handtekening*.



Figuur 4.4: Berichtcodering met PGP

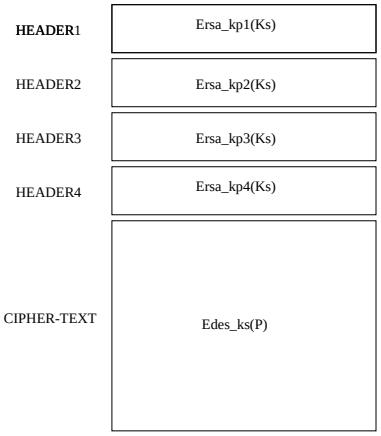
Een andere toepassing van publieke sleutels, *PGP* (Pretty Good Privacy), wordt vooral gebruikt bij email (eenmalige relatief korte berichten van verzender naar ontvanger). In PGP wordt het bericht eerst gecodeerd met een symmetrisch algoritme zoals *IDEA* of *DES* met een random gegeneerde sessiesleutel K_s. In een '*header*', een extra toevoeging voorafgaande aan het gecodeerde bericht, wordt de sessiesleutel K_s gecodeerd met de publieke sleutel van de ontvanger K_p. Deze laatste encryptie geschiedt met een publiek sleutelsysteem zoals *RSA* of *ElGamal*.

De random gegenereerde sessiesleutel K_s wordt afgeleid van een string van 128 (computer)bits. Deze bits ontstaan uit ingevoerde karakters en 32 bits tijdstippen. Deze 128 bits

worden nog een keer gecombineerd met een vorige keer dat er 128 bits zijn gevormd. Tenslotte worden deze 128 bits gebruikt voor de parameters van de symmetrische encryptie.

Het headerprincipe van PGP is te gebruiken als het bericht door meer dan één ontvanger gelezen mag worden. Bij het bericht worden een aantal extra headers geplaatst, elk gecodeerd met de publieke sleutel van één van de ontvangers. Het plaatsen van headers mag alleen geschieden als de totale integriteit van het bericht niet in gevaar komt. Deze methode lost een van de problemen op die met '*escrow*' ontstaan, de overdracht van geheime sleutels aan derden. Een van de headers kan gecodeerd zijn met de publieke sleutel van een overheidsorganisatie die verantwoordelijk is voor de orde en veiligheid. Zij zijn in staat het bericht te decoderen, zonder dat de geheime sleutel van de verzender noodzakelijk is.

Helemaal zonder problemen is deze oplossing niet. Bij een '*perfecte codering*' bestaat natuurlijk de mogelijkheid dat de veiligheidsorganisatie een andere plain-text onder ogen krijgt dan er eigenlijk verzonden is. Omdat de afzender een namaak sessiesleutel kan verstrekken, zal een veiligheidsorganisatie gegarandeerd de juiste sessiesleutel willen ontvangen.



Figuur 4.5: Berichtcodering met multi-PGP

Het is niet noodzakelijk sleutels uit te wisselen. Een symmetrisch sleutelsysteem kan gebruikt worden zonder dat de partijen elkaars sleutel hoeven te kennen. De eerste partij codeert een plain-text met zijn eigen sleutel *Ka*. Daarna wordt een cipher-text verzonden naar de tweede partij. De tweede partij codeert een cipher-text nogmaals met zijn eigen sleutel *Kb* en verzendt het weer terug naar de eerste partij. Deze decodeert het bericht met zijn eigen sleutel *Ka* en verzendt een gedecodeerde cipher-text voor de tweede keer naar de tweede partij. Tenslotte decodeert de tweede partij een cipher-text weer met zijn eigen sleutel *Kb* tot de originele plain-text. Alle partijen passen alleen hun eigen sleutels zodat er geen uitwisseling van sleutels nodig is.

Deze toepassing van symmetrische sleutels stelt een extra eis aan de algoritmen. De encryptie en de decryptie mogen niet verhinderd worden door de encryptie van de andere

partij. In de uitdrukking $D_{Kb}(D_{Ka}(E_{Kb}(E_{Ka}(P)))) = P$ moeten de functies E_{Kb} en E_{Ka} met elkaar verwisseld kunnen worden. Dit betekend dat het commuterende functies moeten zijn.

Een extra probleem ontstaat door de drie verzendingen van hetzelfde bericht in diverse coderingen. Een kraker die het transmissiekanaal afluistert kan de volgende cipher-texten ontvangen: $E_{Ka}(P)$, $E_{Kb}(E_{Ka}(P))$ en $D_{Ka}(E_{Kb}(E_{Ka}(P)))$. Hieruit valt de kritieke informatie $E_{Ka}(P)$ en $E_{Kb}(P)$ af te leiden. Het *discrete logsysteem* van Shamir blijkt bestand tegen deze statistische aanval.

4.6 Elektronische handtekeningen

Voor een *elektronische handtekening* wordt aan het publieke sleutelsysteem een extra voorwaarde gesteld. Publieke sleutels mogen niet alleen gebaseerd zijn op het principe van asymmetrische sleutelsystemen, systemen waarin twee sleutels noodzakelijk zijn. Bovendien moeten sleutels elkaars effecten ongedaan kunnen maken. Met andere woorden, als sleutel $K1$ voor encryptie gebruikt is, dan moet sleutel $K2$ gebruikt worden bij het decoderen en vice versa.

Om een elektronische handtekening te maken, codeert de eerste partij met zijn geheime sleutel $K2$ een plain-text P. Daarna wordt het bericht $E_{K2}(P) = C_1$ opnieuw gecodeerd met de publieke sleutel $K1$ van de tweede partij $E_{K1}(E_{K2}(P)) = C_2$. Het dubbel gecodeerde bericht C_2 wordt na ontvangst door de tweede partij gedecodeerd met decryptiesleutel $K2$ tot een cipher-text $D_{K_2}(E_{K1}(E_{K2}(P))) = (E_{K2}(P)) = C_1$. De tweede partij kan nu controleren of dit bericht van de eerste partij afkomstig is door C_1 te decoderen met de publieke sleutel $K1$ van de eerste partij: $D_{K1}(D_{K2}(E_{K1}(E_{K2}(P)))) = P$. Beide partijen hoeven op deze manier elkaars geheime sleutels niet te kennen. Met zo'n elektronische handtekening voorkomt men spoofing en tampering.

Een bijzondere handtekening is de'*elektronische blinde signering*'. Een blinde signering komt overeen met het signeren van een document zonder de inhoud van het document te kennen. Het zelfde effect verkrijgt men door een document en een stuk carbonpapier in een gesloten enveloppe. Een handtekening op de enveloppe wordt ook op het document geplaatst. De handtekening blijft voorgoed op het document aanwezig. Blinde handtekeningen worden gebruikt in situaties waarin de ondertekenaar moet weten wat voor document het is, maar bepaalde informatie in het document niet mag weten. Meestal is dit een situatie waarin een tussenpersoon een bepaalde handeling op een document moet uitvoeren zonder de volledige inhoud te mogen weten. Bijvoorbeeld een mailserver die een email blind moet signeren en afleveren.

Het is niet zeker of een elektronische handtekening met of zonder certificering de wil tot overeenstemming tussen de partijen aantoont. Bovendien zijn ontvangst en verzendgaranties op Internet niet waterdicht (*repudiatie*). Dit soort juridische onzekerheden en technische onvolmaaktheden zijn ervoor verantwoordelijk dat de belooften van *e-commerce*

op het open Internet (nog?) niet uitkomen. Daarentegen zijn de commerciële activiteiten op Internet tussen partijen die op een andere andere manier elkaars vertrouwen hebben gewonnen, wel toegenomen. Helemaal zonder betekenis is de elektronische handtekening niet. Als er eenmaal een vertrouwensbasis bestaat tussen de partijen, kan de elektronische handtekening - om administratieve procedures te rationaliseren - gebruikt worden als een middel met een beperkte bewijskracht voor het bevestigen van overeenkomsten.

4.7 Toegangsidentificatie

Identificatie maakt gebruik van een encryptie-algoritme waarvoor geen decryptie-algoritme bestaat. Dat wil zeggen, het encryptie-algoritme is een '*éénwegs functie*', een functie die computationeel moeilijk inverteerbaar is (bijvoorbeeld *MD4* en *MD5*). Van een password of een toegangscode wordt een niet meer te decoderen encryptie gemaakt. Deze gecodeerde toegangscode kan zonder kraakgevaar worden opgeslagen in een 'read-only' bestand. Zodra iemand zich identificeert, wordt de gebruikte toegangscode gecodeerd en vervolgens vergeleken met de code in het 'read-only' bestand. De identificatie is geslaagd indien beide coderingen gelijk zijn. In dit password-bestand staan ook andere persoonlijke gegevens van de gebruiker.

Zoals eerder is gezegd, zijn gebruikers onvoorzichtig met de keuze van hun toegangscode. Om het kraken te bemoeilijken zijn volgende maatregelen mogelijk:

- Het verhogen van de entropie door het toevoegen van 'zout' (*salt*) aan een toegangscode. Bijvoorbeeld 10 bit zout geven een toegangscode 4096 extra alternatieven. De 'zout-bits' verhogen de computationele inspanning. Bovendien voorkomt het zout dat twee gebruikers die toevallig dezelfde toegangscode hanteren, elkaars gecodeerde toegangscode als gelijk herkennen. Bestanden van frequente gecodeerde toegangscoden worden door deze zout-variant 4096 keer groter. Tegenwoordig is 10 bit zout onvoldoende voor computationele bescherming;
- De namen en de gecodeerde toegangscodes van de gebruikers worden opgenomen in een '*shadow file*' die alleen door de verantwoordelijke systeembeheerder gelezen en gewijzigd kunnen worden. Andere persoonlijke gegevens die openbaar mogen zijn, blijven in de originele 'read-only' *password file* gehandhaafd. De programma's die toegangscodes in de 'shadow file' moeten verifiëren of wijzigen, krijgen een speciale status (d.m.v. een s-bitje) bij de uitvoering van hun taak. Deze progamma's moeten door de verantwoordelijke systeembeheerder op bijzondere wijze beschermd en beheerd worden zodat zij niet getamperd kunnen worden.

4.8 Certificering

Er zitten aan het publieke sleutelsysteem een paar problemen die nader bekeken moeten worden:

- Als gebuikers hun sleutels kwijtraken, is er een instantie waar zij hun coderingen kunnen herstellen;
- Gebruikers moeten een sleutelpaar verkrijgen dat past bij hun beveiligingsbehoefte. Het genereren van een sleutelpaar moet op een betrouwbare, veilige manier gebeuren. Het genereerproces van zo'n sleutelpaar kan een doelwit van een kraker worden;
- Gebruikers moeten vertrouwen hebben in hun eigen publieke sleutels en die van andere partijen. Een kraker kan zich voordoen als een andere partij en een valse publieke sleutel aanbieden. Bovendien mag een kraker niet in staat zijn de publieke sleutel van een andere partij te wijzigen;
- Als een publieke sleutel verloren is of is veranderd, moeten de andere partijen daarvan op de hoogte gebracht worden, zodat zij niet meer van deze sleutel gebruik zullen maken of berichten accepteren met een ongeldige elektronische handtekening;
- Gebruikers moeten de mogelijkheid hebben hun sleutels veilig op te slaan. Bovendien moeten de publieke sleutels direct en vrijelijk beschikbaar zijn voor legaal gebruik door andere partijen;
- Sleutels mogen maar een beperkte tijd geldig zijn, deze tijd moet aan elke partij duidelijk kenbaar gemaakt worden. Dit kan problemen geven met elektronisch getekende documenten die voor langere tijd bewaard moeten worden.

Om deze beveiligings- en gebruiksproblemen te voorkomen kan men gebruik maken van gecertificeerde vertrouwenspartijen. Dit zouden openbaar gecontroleerde organisaties kunnen zijn die door alle partijen vertrouwd moeten worden. Toch zijn er behoorlijke bezwaren tegen deze oplossing. De belangrijkste nadelen van het afgeven van een sleutel, *key escrow,* aan een gecertificeerde organisatie zijn de volgende:

- Alle cryptografische toepassingen zijn in principe zonder key escrow mogelijk. Er zijn methoden die geen sleutel nodig hebben zoals *steganografie*. Dit zijn methoden waarbij het bericht verstopt wordt tussen - of in - andere berichten.
- De handelingen bij het deponeren kunnen het doelwit worden van een aanval. De opgeslagen sleutels kunnen aangevallen worden. Medewerkers van de 'escrow organisation' kunnen omgekocht worden. Het is ook mogelijk dat de sleutels door

een ‘escrow organisation’ misbruikt worden. Bovendien is certificering niet altijd een garantie voor zekerheid en betrouwbaarheid, zeker als de ‘escrow organisation’ gecontroleerd wordt door onbetrouwbare instanties;

- Hoe zou je ooit via een mogelijk onbetrouwbaar kanaal zoals Internet en een mogelijk onbetrouwbare tussenpartij vertrouwen kunnen krijgen in een onbekende partij? Mensen blijken uiteindelijk toch terug te vallen op het persoonlijke contact om vertrouwen in elkaar te krijgen.

De nadelen van ‘key escrow’ kunnen gedeeltelijk voorkomen worden door de sleutel in gedeelten aan verschillende instanties af te geven.

4.9 Elektronische verkiezingen

Elektronische verkiezingen komen steeds meer in zwang. Niet alleen bij de verkiezing van de beste webpagina, maar ook bij de officiële verkiezing van een politieke vertegenwoordiging. De meeste elektronische verkiezingen zijn nogal fraudegevoelig of maken inbreuk op de privacy van de kiezers. Als elektronische verkiezingen niet het vertrouwen van het publiek willen verliezen, moeten zij aan een aantal eisen voldoen:

Democratisch: . Een elektronische verkiezing is democratisch als alleen kiesgerechtigden kunnen stemmen. Bovendien moet het garanderen dat elke kiesgerechtigde maar één keer kan stemmen;

Bescherming van de privacy: Een elektronische verkiezing beschermt de privacy van de deelnemers als niemand in staat is de stem van een kiezer te achterhalen. Bij elke verkiezing bestaat de kans op het chanteren en het omkopen van kiezers. Daarom moet zoveel mogelijk voorkomen worden dat de kiezer zijn stem kan bewijzen;

Controleerbaar: Een elektronische verkiezing is controleerbaar als iedereen onafhankelijk kan controleren dat het verkiezingsresultaat correct is. Een zwakkere eis is, als kiezers in staat zijn hun eigen stem te controleren en eventuele fouten tijdens het stemmen te corrigeren.

Laagdrempelig: Een elektronische verkiezing is laagdrempelig als kiezers gemakkelijk hun stem uit kunnen brengen, zonder dat bijzondere vaardigheden en apparatuur noodzakelijk zijn.

Flexibel: Een elektronische verkiezing is flexibel als het geschikt is voor open- en gesloten vragen. Kwantitatieve- en kwalitatieve onderzoeksvragen moeten mogelijk zijn. Bij de keuze voor een cipher-systeem moet men daarmee rekening houden.

Niet plaats gebonden: . Een elektronische stemming is niet-plaats gebonden als men op willekeurige plaatsen kan stemmen. De meeste kiezers zien dit als het belangrijkste voordeel van elektronische verkiezingen waardoor de participatiegraad zal stijgen. Anderzijds verhogen niet-plaats gebonden verkiezingen wel de kans op fraude en chantage.

Een eenvoudige elektronische verkiezing zou op de volgende manier kunnen plaats vinden: In eerste instantie stuurt een kiezer zijn keuze en zijn identificatie aan een validator. Deze validator kan de identificatie gebruiken om de kiezer af te turven op de lijst met geregistreerde kiezers. Vervolgens wordt de identificatie verwijderd en de keuze doorgegeven aan de teller. Tenslotte wordt de totaalstand gepresenteerd.

In deze aanpak zijn een paar fraudemogelijkheden aanwezig. Iemand kan zich voordoen als een geregistreerde kiezer. Bovendien zijn de kiezers niet zeker of hun privacy beschermd wordt. Tenslotte kan de validator een andere keuze doorgeven aan de teller.

De kiezer codeert met de publieke sleutel van de teller zijn stem. Vervolgens codeert hij deze gecodeerde stem nogmaals met zijn eigen geheime sleutel. Na ontvangst door de validator decodeert deze het bericht met de publieke sleutel van de kiezer, waarmee de identiteit van de kiezer eenduidig vastgesteld wordt. Vervolgens wordt met de publieke sleutel van de teller gecodeerde stem doorgezonden naar de teller. Deze decodeert de keuze met zijn eigen geheime sleutel. Met dit coderingsschema wordt voorkomen dat de kiezer zich als een ander voor kan doen en dat de validator de stem van de kiezer kan lezen of wijzigen. Pas als twee of meer partijen samenwerken is fraude mogelijk.

Een van de manieren om de samenwerking van de validator en de teller voorkomen, is het gebruik van een blinde handtekening tussen validator en teller. De kiezer codeert zijn stem en koppelt dit aan zijn identificatiecode. De gecodeerde stem met identificatiecode wordt verzonden naar de validator. Deze turft de stemgerechtigde kiezer en verstuurt de gecodeerde stem terug naar de kiezer met een blinde handtekening van de validator. De kiezer zendt zijn gecodeerde stem met de blinde handtekening van de validator naar de teller. De teller controleert de blinde handtekening van de validator en verwerkt de stem.

4.10 Opgaven

1. Probeer het volgende Ceasargecodeerde bericht te ontcijferen:

   LNNZDPLNCDJHQLNRYHUZRQ

2. Geef enkele voorbeelden waaruit blijkt dat berichten in het algemeen niet uniform verdeeld zijn. Welke oplossing is geschikt om deze berichten uniform verdeeld te maken?

3. Een andere manier om geheime boodschappen te versturen is steganografie.

(a) Wanneer zouden partijen steganografie gebruiken?

(b) Wat is het nadeel van steganografie?

4. Een bekende manier om geheime boodschappen te ontcijferen is gebruik te maken van letterfrequenties. Deze methode werkt bij systemen waarbij de letters simpelweg vervangen worden door andere tekens, zoals monoalfabetische substituering. Methoden die gebruik maken van verwisselingen van letterposities zijn minder kwetsbaar voor deze methoden.

   (a) Welke principes zijn volgens Shannon noodzakelijk voor een betrouwbaar cryptografisch systeem?

   (b) Als de letters uniform verdeeld zijn in een bericht dan hebben alle letters $i = 1 \ldots 26$ evenveel kans om op te treden $P(X = a) = p_a = 1/26$. Indien wij een ander bericht van gelijke lengte met willekeurig verdeelde letters op het eerste bericht leggen, dan is de kans dat op een positie twee letters 'a' op elkaar liggen gelijk aan $P(X = a \cap X = a) = p_a^2 = (1/26)^2 = 0,0385$. Als wij deze methode per taal uitvoeren, blijkt dat deze coïncidentie per letter per taal verschilt. Om deze eigenschap van een taal met een kental te beschrijven wordt zij gedefinieerd als de *coïncidentie-index*: $I_c = \sum_{i=1}^{26} p_i^2$:

taal	I_c
Engels	$0,0661$
Frans	$0,0778$
Duits	$0,0762$
Italiaans	$0,0738$
Japans	$0,0819$
Russisch	$0,0529$
random	$0,0385$

Tabel 4.1: De coïncidentie-index I_c.

   Bereken de coïncidentie-index voor de Nederlandse taal. Maak gebruik van de gegeven letterfrequenties in bijlage C.

   (c) Hoe zou de coïncidentie-index I_c gebruikt kunnen worden bij het kraken van een cipher-text?

5. Waarom moet de entropie $H(K|C)$ zo groot mogelijk zijn?

6. Een natuurlijke tekst in het Nederlands heeft een relatieve nulde-orde redundantie van 50%. De Nederlandse tekst wordt op karakterbasis met een Ceasarcode versleuteld.

   (a) Bereken de kritieke lengte van de cipher-text.

(b) Indien de Nederlandse tekst gecomprimeerd wordt met een code-efficiëntie van 70%, wat is dan de kritieke lengte van de Ceasar codering?

7. Welke problemen kent het sleutelbeheer?

8. Verklaar het principe van de elektronische handtekening met RSA.

9. Geef de voor- en nadelen van certificering. Geef van de volgende organisaties aan hoe hun standpunt t.a.v. certificering zou luiden:

   - Ministerie van Economische Zaken;
   - Ministerie van Binnenlandse Zaken;
   - Ministerie van Justitie;
   - Ministerie van Defensie;
   - De kamers van koophandel;
   - De vrije software vereniging;
   - Banken;
   - Verzekeringen;
   - Dot.com bedrijven.

10. Wat zijn de voor en nadelen van een elektronische handtekening in het kader van e-commerce?

11. Welke eisen stelt men aan een elektronische verkiezingen?

Bijlage A

Markov analyse met Maple

```
> with(linalg);
> M := matrix([[-1, 1/2, 1/2], [1/5, -4/5, 0], [4/5, 3/10, -1/2],
> [1, 1, 1]]);
```

$$M := \begin{bmatrix} -1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{5} & \frac{-4}{5} & 0 \\ \frac{4}{5} & \frac{3}{10} & \frac{-1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

```
p:=linsolve(M, vector([0,0,0,1]));
```

$$p := \left[\frac{1}{3}, \frac{1}{12}, \frac{7}{12}\right]$$

```
> H:=evalf(evalm(evalm(map(-ln/ln(2),p))&* p));
```

$$H := 1.280672130$$

Bijlage B

De ongelijkheid van Chebyshev

Gegeven een stochast Y met een waardenbereik $0 \leq y$ en een eindig gemiddelde $E(Y)$. Op elke waarde y is de kansfunctie $p(y)$ gedefinieerd. Voor $0 < \delta$ geldt de volgende hulpstelling:

$$P(Y > \delta) \quad < \quad \frac{E(Y)}{\delta}$$

Bewijs hulpstelling:

$$P(Y > \delta) \quad = \quad \sum_{\delta<y} p(y) \quad < \quad \overbrace{\sum_{\delta<y} \frac{y}{\delta} p(y)}^{0<\delta<y} \quad \leq \quad \overbrace{\frac{1}{\delta} \sum_{y} y \cdot p(y)}^{0\leq y} \quad = \quad \frac{E(Y)}{\delta}$$

Wij kunnen deze hulpstelling gebruiken voor een willekeurige stochast X die een eindig gemiddelde $E(X)$ heeft. Als Y gesubstitueerd wordt met $0 \leq |X - E(X)| = \sqrt{(X - E(X))^2}$ dan volgt de *ongelijkheid van Chebyshev*:

$$P(|X - E(X)| > \delta) \quad = \quad P((X - E(X))^2 > \delta^2) \quad < \quad \frac{E((X - E(X))^2)}{\delta^2} \quad = \quad \frac{Var(X)}{\delta^2}$$

De ongelijkheid van Chebyshev wordt ook wel de *zwakke wet van grote aantallen* genoemd.

$$P(|X - E(X)| > \delta) \quad < \quad \frac{Var(X)}{\delta^2}$$

Bijlage C

Prefixcodes

C.1 Shannon-Fanocode voor Nederlandse letters

i	symbool	p_i	$\sum_{j=1}^{i} p_j$	Fano	L_i	$p_i \cdot L_i$
1	E	0,190	0,190	00	2	0,380
2	N	0,110	0,300	0100	4	0,440
3	A	0,066	0,366	0101	4	0,264
4	T	0,065	0,431	0110	4	0,260
5	D	0,063	0,494	0111	4	0,252
6	O	0,063	0,557	1000	4	0,252
7	R	0,060	0,617	1001	4	0,240
8	I	0,054	0,671	1010	4	0,216
9	L	0,042	0,713	1011	4	0,168
10	S	0,041	0,754	11000	5	0,205
11	G	0,038	0,792	11001	5	0,190
12	H	0,026	0,818	11010	5	0,130
13	V	0,025	0,843	110110	6	0,150
14	U	0,024	0,867	110111	6	0,144
15	K	0,020	0,887	111000	6	0,120
16	M	0,020	0,907	111001	6	0,120
17	B	0,015	0,922	111010	6	0,090
18	W	0,015	0,937	111011	6	0,090
19	IJ	0,015	0,952	111100	6	0,090
20	C	0,013	0,965	111101	6	0,078
21	P	0,013	0,978	111110	6	0,078
22	F	0,010	0,988	1111110	7	0,070
23	Z	0,010	0,998	11111110	8	0,080
24	J	0,001	0,999	11111110	8	0,008
25	Q	0,001	1,000	111111110	9	0,009
26	Z	0,000	1,000	$>$111111111	$>$9	0,000
gemiddelde lengte van de codewoorden $E(L)$						4,124

C.2 Huffmancode voor Nederlandse letters

i	symbool	p_i	Huffman	L_i	$p_i \cdot L_i$
1	E	0,190	111	3	0,570
2	N	0,110	011	3	0,330
3	A	0,066	1010	4	0,264
4	T	0,065	1000	4	0,260
5	D	0,063	0111	4	0,252
6	O	0,063	0110	4	0,252
7	R	0,060	0101	4	0,240
8	I	0,054	0100	4	0,216
9	L	0,042	0001	4	0,168
10	S	0,041	10111	5	0,205
11	G	0,038	10011	5	0,190
12	H	0,026	11011	5	0,130
13	V	0,025	11010	5	0,125
14	U	0,024	11000	5	0,120
15	K	0,020	00000	5	0,100
16	M	0,020	101101	6	0,120
17	B	0,015	101100	6	0,090
18	W	0,015	100101	6	0,090
19	IJ	0,015	100100	6	0,090
20	C	0,013	110011	6	0,078
21	P	0,013	110010	6	0,078
22	F	0,010	000010	6	0,060
23	Z	0,010	0000111	7	0,070
24	J	0,001	00001101	8	0,008
25	Q	0,001	000011001	9	0,009
26	X	0,000	000011000	9	0,000
gemiddelde lengte van de codewoorden $E(L)$					4,115

Bijlage D

Nederlandse woorden

In teksten		
lengte	r.f.	c.r.f
1	0,001	0,001
2	0,193	0,194
3	0,216	0,408
4	0,109	0,517
5	0,075	0,592
6	0,094	0,686
7	0,064	0,750
8	0,056	0,806
9	0,050	0,856
10	0,036	0,892
11	0,030	0,922
12	0,018	0,940
13	0,011	0,951
14	0,010	0,961
15	0,005	0,966
16	0,004	0,970
>16	0,003	1,000

Groene boekje		
lengte	r.f.	c.r.f
1	0,0000	0,0000
2	0,0003	0,0003
3	0,0032	0,0035
4	0,0108	0,0146
5	0,0220	0,0366
6	0,0472	0,0838
7	0,0746	0,1584
8	0,1027	0,2611
9	0,1271	0,3882
10	0,1365	0,5247
11	0,1228	0,6475
12	0,1000	0,7475
13	0,0766	0,8241
14	0,0558	0,8799
15	0,0377	0,9176
16	0,0264	0,9440
>16	0,0562	1,0000

Meest voorkomend		
rang	woord	r.f.
1	de	0,074
2	van	0,040
3	het	0,035
4	een	0,027
5	in	0,024
6	en	0,020
7	dat	0,016
8	te	0,014
9	is	0,012

Bijlage E

Eigenschappen van enkele bekende CRC's

CRC-6

$$x^6 + x^5 + x + 1 = (x^4 + x^3 + x^2 + x + 1) * (x + 1)^2$$

De Miminimum Hamming Distance $MHD = 4$. De CRC-6 detecteert:

- 100% één-bit fouten;
- 100% twee-bit fouten;
- 100% oneven-aantal-bit fouten;
- 100% *bursts* van 6 bit of minder;
- $96,875$% bursts 7 bit;
- $98,437$% bursts langer dan 7 bit.

CRC-16 gedefinieerd in de *CCITT V.41* standaard:

$$x^{16} + x^{15} + x^5 + 1 = (x + 1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1)$$

De Miminimum Hamming Distance $MHD = 4$. De CRC-16 detecteert:

- 100% één-bit fouten;
- 100% twee-bit fouten;
- 100% oneven-aantal-bit fouten;
- 100% *bursts* van 16 bit of minder;
- 99.997% bursts 17 bit;

- 99.998% bursts langer dan 17 bit.

CRC-32 gedefinieerd in de *IEEE 802.3* standaard als de *priempolynoom* (niet in factoren te ontbinden):

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

CRC-32 heeft een Minimum Hamming Distance $MHD = 4$ als zij gebruikt wordt voor berichten van minder dan 12,144 bit informatie (inclusief de 32 bits redundantie). Als de lengte van het bericht beperkt wordt tot $1,632$ bit informatie (inclusief redundantie) is de Mimimum Hamming Distance 5.

De CRC-32 detecteert:

- 100% één-bit fouten;
- 100% twee-bit fouten;
- 100% drie-bit fouten;
- 100% vier- bit fouten;
- 100% *bursts* van 32 bit of minder;
- 99.999999953% bursts 33 bit;
- 99.999999976% bursts langer dan 33 bit.

De CRC-32 is niet gevoelig voor oneven-aantal-bit fouten.

Literatuur

[1] D. Applebaum: *Probability and Information*, Cambridge University Press, 1996

[2] D.E. Boekee: *Informatietheorie I, Vraagstukken*, Collegedictaat Laboratorium Informatietheorie, afd. Electrotechniek TH-Delft, 1969-1970

[3] IJ. Boxma: *Informatietheorie I*, Collegedictaat Laboratorium Informatietheorie, afd. Electrotechniek TH-Delft, 1969-1970

[4] J.C.A. van der Lubbe: *Basismethoden cryptografie*, Delftse Universitaire Pers, 1997

[5] C.E. Shannon: *A Mathematical Theory of Communication*, The Bell System Technical Journal, Vol. 27, pp. 379 423, 623 656, July, October, 1948

[6] A. Sinkov: *Elementary Cryptanalysis*, Mathematical Association of America, New Mathematical Library, 1966

[7] W. Stallings: *Netwerkbeveiliging en cryptografie*, Academic Service, Schoonhoven (2000)

[8] M.J. Usher: *Information Theory for Information Technologists*, Macmillan Computer Science Series, 1984

Index