LINEÆR KOMPLEKSITET TIL PRODUKTER AV MAKSIMALSEKVENSER

Torbjørn Ovnerud
Institutt for Informatikk, Universitetet i Bergen
tobben@ii.uib.no

8. november 2000

Forord

Når dette arbeidet nå er fullført vil jeg i hovedsak rette en takk til min veileder, Tor Helleseth. Vi har i samarbeid kommet fram til resultatene som blir presentert her, og uten den veiledning han har gitt ville det nok ikke blitt noe arbeid ut av det. Han har alltid vært tilgjengelig når jeg har hatt behov for assistanse

I tillegg vil jeg takke mine venner for gode utenomfaglige samtaler på pauserommet. De har vært med på å skape et studiested som har vært til å overleve.....:)

Innhold

1 Innledning **6**

1.1 Kryptografi-introduksjon 6

1.1.1 Klassifisering 7

1.1.2 One-time pad 9

1.2 Strømchiffer 9

1.2.1 Nøkkelgeneratoren 10

1.3 Sekvenser 10

1.3.1 Maksimalsekvenser 10

1.3.2 Lineær-kompleksitet og tilfeldighet 11

1.4 Sikkerhet til strømchiffer 11

1.5 Om oppgaven 12

1.5.1 Utførelse 12

1.5.2 Oppbygning 13

1.5.3 Resultat 13

2 Teoretisk bakgrunn **14**

2.1 Elementære fakta 14

2.2 Skiftregistre 15

2.2.1 Mer teori 16

2.3 Tracefunksjonen 17

2.3.1 Eksempel 18

2.3.2 Flere resultater 19

2.4 Keys resultat 20

2.5 Teoremer 21

2.5.1 Eksempel 23

2.6 Berlekamp-Massey 24

3 Gjennomføring **26**

3.1 Om oppbyggingen 26

3.2 Behandling av data 28

3.3 Resultater 29
3.3.1 Polynomer med sammensatt grad 29
3.3.2 Polynomer med primtalls grad 30

4 Resultatanalyse **32**
4.1 Maksimal reduksjon 32
4.2 Ekvidistante skift 34
4.2.1 Tilfelle med vekt lik to 35
4.2.2 Tilfelle med vekt lik en 37
4.3 Andre tilfeller 39
4.4 Videre forskning 40
4.4.1 Bevis av nedre grense 40
4.4.2 Se på flere skift 41

5 Oppsummering og konklusjon **42**

A Programkode **44**
A.1 Hovedprogrammet 44
A.2 Berlekamp-Massey algoritme 47

B Tabeller **50**
B.1 Starttilstander 50
B.2 Iterasjonsverdier 50
B.3 Faktorisering av $2^m - 1$ 51
B.4 Resultattabeller 51
B.4.1 Grad lik 3 52
B.4.2 Grad lik 5 52
B.4.3 Grad lik 7 53

Figurer

1.1 Kommunikasjonskanal 7
1.2 Klassifisering 8
1.3 Generell pseudo-tilfeldig chiffersystem 9

2.1 Enkelt skiftregister 16

3.1 Kjøretid 27
3.2 Programmet 28

Algoritmer

1 Berlekamp-Massey 25

Kapittel 1

Innledning

Helt siden oldtiden har ønsket om å holde informasjon hemmelig vært en viktig del av menneskets hverdag. Man har prøvd å beskytte seg mot uvedkommendes innsyn i ens private anliggender, eller innsyn i den informasjon du måtte ha om dem selv. Etter datamaskinens inntog og med dens mulighet for å sende store mengder data over telefonlinjer og andre media, har utviklingen gått mer og mer hurtig. Systemer har kommet og forsvunnet, noen har blitt standardisert og andre er i prototypestadiet. Man har gått fra enkle substitusjonssystemer, til kompliserte og beregningstunge algoritmer hvor en gjør bruk av offentlige nøkler. Felles for dem alle er det faktum at menneskets trang for å skjule og å gjemme bort er stort.

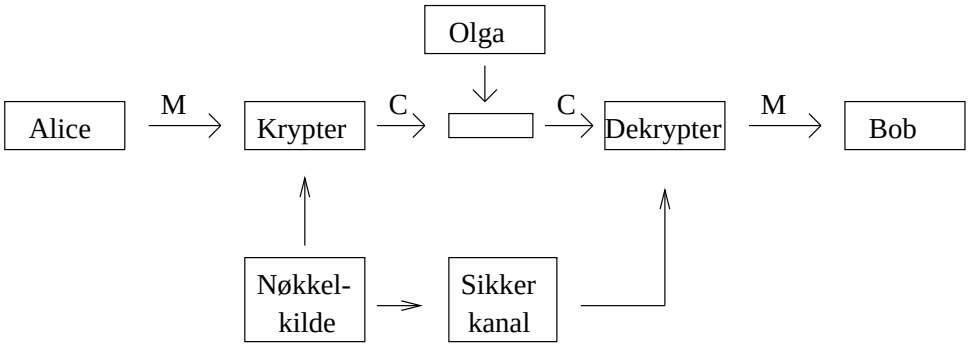
Hvorfor man vil skjule ting for andre kan ha mange årsaker. For privatpersoner er retten til privatliv et viktig moment. E-post, nettbank og mobiltelefoni er vanligvis ikke noe en vil kringkaste, men heller holde for seg selv. Banker, forsikringsselskaper, netthandelsplasser og andre større institusjoner er opptatt av at sensitiv informasjon som kontonummer, pengeoverførsler og lignende ikke skal komme på avveie. Nasjoner er opptatt av at forsvarshemmeligheter som utveksles ikke skal kunne fanges opp av en potensiell fiende.

I de siste årene med økende bruk av elektroniske tjenester over usikrede linjer via Internett, er det også problemer med ektheten, avsenderidentitet og muligheten for mennesker med kunnskap om emnet å snappe opp det du sender. Det faktum at regjeringer er villige til å investere i utstyr som gjør det mulig å overvåke og snappe opp meldinger over hele kloden, gjør folk mer nervøse og villige til å sette seg inn i og investere i kryptoteknologi.

1.1 Kryptografi-introduksjon

Kryptografi brukes idag for å beskytte, bevare, holde hemmelige og signere dokumenter, filer eller annet som en sender over en kanal hvor muligheten for avlytting eller

korrumpering er tilstedeværende. Scenarioet en vanligvis kommer borti er som følger; En har en sender, Alice, en mottaker, Bob, og en mulig inntrenger Olga. Alice og Bob kommuniserer over en usikker kanal og vil derfor kryptere sine meldinger. De vil hindre Olga i å forstå og eventuelt endre meldingene deres. Et eksempel på dette scenarioet kan sees i figuren over en kommunikasjonskanal (Fig 1.1)



Figur 1.1: Kommunikasjonskanal

Her vil Alice sende en melding, M, til Bob. Hun bruker sin nøkkel og krypteringsfunksjonen på M, og resultatet hun sender er chifferteksten, C. Olga vil gjerne vite hva Alice sender, men hun kan ikke finne det ut. Figuren beskriver en situasjon hvor en bruker hemmelig nøkkel kryptografi. Man må på forhånd ha utvekslet en nøkkel til bruk i kryterings- og dekrypteringsfunksjonen. Alternativet er offentlig nøkkel kryptografi, hvor Alice, og alle andre, har en nøkkel hun kan kryptere meldinger til Bob med. Bob er i besittelse av en annen nøkkel som gjør det mulig for ham, og bare ham, å dekryptere meldingen. Meldingen er da ikke-invertibel for andre enn Bob. Slike systemer baserer seg på uløste matematiske problemer.

I dette scenarioet kan en legge inn autentisering, signering eller andre opsjoner for å sikre integriteten til meldingen som en sender. Dette gjør det mulig for etterhvert å kunne bruke elektroniske dokumenter som er garantert å være ekte og å komme fra den rette avsender.

For at et sett av personer skal kunne kommunisere med hverandre på en sikker og pålitelig måte, er det nødvendig at disse blir enige om et kryptosystem. Figuren over (Fig 1.1), viser også et et kryptosystem, uten at det her er tatt hensyn til hvilken algoritme som brukes.

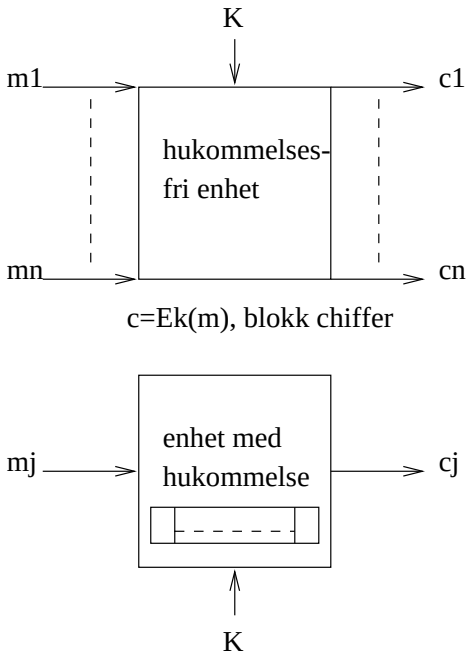
1.1.1 Klassifisering

En kan dele opp kryptografiske systemer i to store hoveddeler. Blokkchiffer og strømchiffer. Begge har sine bruksområder, hvor de gir det beste resultatet. Ved blokkchiffer deler man opp klarteksten i blokker av lik lengde, ofte ≥ 64 bit, som man så sender

igjennom en funksjon, bruker nøkkelen og som ved utgangen av funksjonen er kryptert. Blokkchiffer er egentlig enkle substitusjonschiffer, men som ved mange runder med kryptering og med et stort nøkkelalfabet, ofte gjør det praktisk umulig å finne klarteksten hvis man ikke kjenner den korresponderende nøkkelen. Nøkkelen vil og være nærmest umulig å finne da nøkkelalfabetet ofte vil være > 64 bit. Funksjonen som er i bruk gjentas for alle blokker, og sådan er rene blokkchiffer hukommelsesfrie, krypteringen av en blokk er upåvirket av krypteringen til alle andre blokker.

Strømchiffer deler opp meldingen i tegn eller bits og krypterer hver og en av dem av gangen ved hjelp av en tidsvarierende funksjon. Tidsavhengigheten er bestemt av den interne statusen til strømchifferen. Denne tidsavhengigheten sammen med en regel for hvordan tilstanden til strømchifferen skal forandre seg, gjør at kryperingen av to like klarteksttegn som oftest vil være forskjellige. I strømchiffer vil den neste tilstanden til nøkkelen være avhengig av den forrige. Ved denne konstruksjonen er det sagt at strømchiffer har hukommelse.

Denne klassifiseringen av de to forskjellige oppbyggingene av kryptosystem er klar analogt til inndelingen av feilkorrigerende koder i blokkoder og konvolusjonskoder. Klassifiseringen er illustrert i figur (1.2) under.



Figur 1.2: Klassifisering

1.1.2 One-time pad

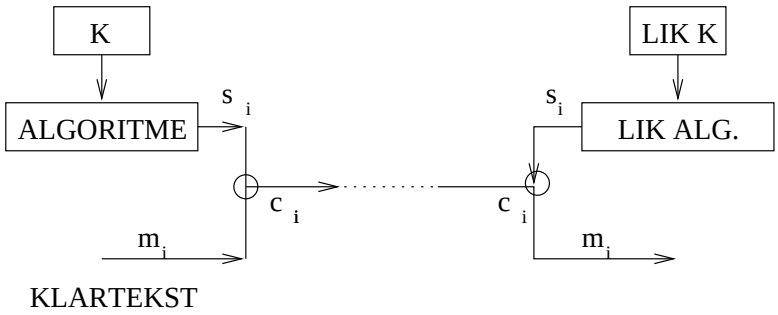
Et kryptosystem som innehar egenskapen av å være perfekt, ved å anta at man bruker et rent chiffertekst-angrep[1], er One-time pad. Det er et meget enkelt system som egentlig bare adderer bit for bit eller tegn for tegn fra nøkkelstrengen og meldingsstrengen sammen $\mathrm{mod}2$, eller

$$c_i = m_i \oplus k_i,\ for\, i = 1, 2, 3 \ldots$$

Dekrypteringen vil bare være å addere chifferteksten med nøkkelen igjen, og klarteksten vil falle ut.

Sikkerheten for dette systemet baserer seg på at nøkkelstrengen er en *helt* tilfeldig og ikke gjentar seg. Ved å bruke en helt tilfeldig nøkkelstreng kan ikke en eventuell angriper skille den rette klarteksten fra hvilken som helst annen meningsfull klartekst. Dette gjør at en ikke kan angripe systemet statistisk.

Ulempen, som er stor, er at en må ha tilgang på en uendelig, eller i det minste like lang som meldingen, nøkkelstrøm. Den skal i tillegg være helt tilfeldig, og den kan ikke gjentas. Dette er vanskelig å oppnå. I tillegg må nøkkelen distribueres på en sikker måte. Egenskapene er allikevel så appelerende at det leder en om å bygge strømchiffer på dette prinsippet, hvor en istedenfor bruker en pseudo-tilfeldig sekvens som nøkkel. Se figur (1.3) under for en generell beskrivelse.



Figur 1.3: Generell pseudo-tilfeldig chiffersystem

1.2 Strømchiffer

Strømchiffer spesifiserer en enhet med internminne som transformerer en klartekstbit til en chiffertekstbit ved hjelp av en funksjon som avhenger både av nøkkelen og den interne tilstanden til strømchifferet. Strømchiffer kan deles opp i to deler. I *synkrone* strømchiffer er den neste tilstanden bare avhengig av den forrige tilstanden og ikke

[1] Angriper har bare tilgang på chifferteksten

av input. Dette gjør transformasjonen hukommelsesfri, men tidsavhengig. En pleier da å skille nøkkelgenereringsdelen og krypteringsdelen fra hverandre. I *selv-synkrone* strømchiffer bruker en i krypteringen og dekrypteringen en endelig mengde minne for å kunne rette opp tap av tegn og annet for igjen å synkronisere sender og mottaker.

1.2.1 Nøkkelgeneratoren

I synkrone strømchiffer ser en generelt på nøkkelstrømsgeneratoren som en endelig tilstandsmaskin. En har et utputtalfabet og et sett av tilstander. En har en tilstandsfunksjon som bestemmer den neste tilstanden ut fra den forrige, og en har en utputtfunksjon som bestemmer utputtbiten fra den nåværende tilstanden. Som nøkkelgenerator bruker en svært ofte et *Lineært Feedback SkiftRegister*, eller et LFSR. Årsaken for dette er at de er lette å implementere i hardware, de kan produsere lange pseudo-tilfeldige strenger sammen med gode statistiske egenskaper og de kan lett analyseres. Med dette menes at en kan lett finne egenskaper som gjør dem bedre i bruk.

Et slikt skiftregister består av registeret selv og av en feedback-funksjon. Denne funksjonen er definert av et passende polynom. Registeret er en sekvens av bits som blir skiftet et steg og en genererer en ny ved hjelp av funksjonen. Funksjonen er en bitvis XOR mellom deler av innholdet i skiftregisteret. Hvilke biter som brukes av funksjonen er igjen bestemt av feedbackpolynomet.

Ved bruk av et slikt system som her er beskrevet vil en etter en gitt tid få en gjentagelse av utputtbitene. Det er da viktig å sørge for lengst mulig periode før en gjentagelse forekomme.

1.3 Sekvenser

Strømchiffer bruker deterministisk genererte pseudotilfeldige sekvenser for å kryptere klarteksten. Disse sekvensene er lange strømmer med tall definert innenfor et visst alfabet, som for eksempel det vi skal konsentrere oss om, $GF[2]$. Grunnet det faktum at nøkkelgeneratoren er en endelig tilstandsmaskin, vil nødvendigvis nøkkelstrømmen være periodisk, og denne perioden vil være endelig. En er da stilt overfor to problemstillinger, størst mulig periode og best mulig statistiske egenskaper.

1.3.1 Maksimalsekvenser

De sekvensene som en oftest ønsker å bruke i strømchiffer er såkalte *maksimalsekvenser*, eller *m-sekvenser*. Disse sekvensene innehar den egenskap at de her maksimal periode i forhold til størrelsen eller lengden på sin generator. I tillegg har m-sekvenser en del gode statistiske egenskaper i forhold til tilfeldighet. det vil si de tilfredsstiller Golombs tilfeldighets postulater. Disse er listet i [1], s 180.

Disse postulatene gir oss krav til sekvensene våre, men de gir ikke et mål på tilfeldigheten til en sekvens. De beskriver så å si bare sekvenser hvor skiftregisteret har en feedback-funksjon definert av et primitivt polynom.

1.3.2 Lineær-kompleksitet og tilfeldighet

For videre å kunne undersøke sekvensene for hvilken grad av tilfeldighet de har, må en finne et mål for dette. Forskjellige metoder for dette har vært foreslått, men en akseptert måleenhet er $L(s)$, *Den lineære kompleksiteten* til den uendelige sekvensen s. $L(s)$ er definert som følger:

1. Hvis s er nullsekvensen, er $L(s) = 0$.
2. Hvis ingen LFSR genererer s, er $L(s) = \infty$.
3. I de resterende tilfeller er $L(s)$ lik lengden på det korteste LFSR som kan generere s.

I vårt tilfelle er det kun snakk om endelige binære sekvenser og kun punkt 1 og 3 er av interesse. Å bruke dette målet er særlig appelerende siden en har gode metoder for å finne denne verdien. For en endelig, tilfeldig sekvens av lengde n vil den lineære kompleksiteten i de fleste tilfeller ligge rundt $n/2$ [2]. For m-sekvenser er denne verdien lik graden til feedbackpolynomet.

Lineær-kompleksitet er viktig da det eksisterer metoder for å analysere sekvenser med hensyn på dette. Denne bør være stor da en kan bruke algoritmer som raskt kan definere et skiftregister som kan generere sekvensen. I tillegg bør sekvensen vi ser på ha en typisk lineær-kompleksitetsprofil som ligger rundt $n/2$ linjen etterhvert som sekvensen vokser. Med dette menes at sekvensen bestående av 62 påfølgende 0-ere og en 1-er har høy lineær kompleksitet, men ikke er særlig tilfeldig. Så en høy lineær-kompleksitet gir nødvendigvis ikke en god generator, men lav gir ihvertfall en dårlig. En mer inngående analyse av forventning og fordeling av $L(s)$ finnes i Rueppel [2].

Mer teori og bakgrunn vil komme i neste kapittel.

1.4 Sikkerhet til strømchiffer

Som vi så i forrige avsnitt har en enkle grep en kan foreta seg for å gi sekvensene en vil generere gode egenskaper. Allikevel er ikke en enkel generator med gode egenskaper nok for å sikre et sikkert strømchiffer system. En mulig løsning er å bruke ikke-lineær feedback. Problemet med disse er at de er vanskelig å analysere, og de har en del uønskede egenskaper som kort periode. En annen løsning blir da å kombinere utputten fra flere slike skiftregistre på en ikke-lineær måte slik at resultatsekvensen får en større

lineær-kompleksitet. En bruker en ikke lineær funksjon som filtrerer og utfører operasjoner på sekvensene fra disse registrene og gir dem ikke-lineæritet. Slike måter er blant annet sekvensmultiplikasjon og sekvensaddisjon.

Ved å utføre denne form for operasjoner vil en kunne øke verdien av den lineære-kompleksiteten drastisk. Grunnen til at en vil gjøre dette er at en kan beholde en rimelig kort lengde på skiftregistrene sine, og allikevel få bedre sekvenser statistisk sett. En vil allikevel ikke få lengre perioder men, bedre generelle egenskaper. Årsaken til dette er at en fjerner en del av de lineære sammenhengene en har mellom påfølgende bits i en enkelt sekvens og gjør den vanskeligere å analysere for en angriper.

Vi vil videre i arbeidet se spesielt på multiplikasjon av maksimalsekvenser og deres tilhørende lineær kompleksitet. Ulempen med denne multiplikasjonen er at en annen viktig egenskap med sekvensene faller bort. Fordelingen av 0-ere og 1-ere blir meget skjev. For produkt av n sekvenser vil antall 1-ere som regel være omtrent $1/2^n$ når fordelingen i hver sekvens er jamn. Vi har allikevel valgt å ikke bekymre oss om dette ettersom det finnes gode metoder for å rette opp problemet.

1.5 Om oppgaven

Som tittelen på dette avsnittet avslører, vil vi her gi en kort gjennomgang av innholdet i denne teksten og presentere kort hva vi mener har kommet fram av dette.

1.5.1 Utførelse

Målet med oppgaven er å se på den lineære kompleksiteten til produkt av ulike skift av maksimalsekvenser. Tidligere forskning har gitt resultater med hensyn på produkt av original sekvens og et tilfeldig skift av denne sekvensen, og med hensyn på produkt av vilkårlig antall ekvidistante skift av en maksimalsekvens. Med ekvidistante skift menes at skiftene er på formen $k\delta$ der δ er lengden på det første skiftet. Formålet med vår forskning er å se spesielt på produkt av tre slike skift som ikke nødvendigvis var ekvidistante. Ønsket er å analysere hvilke skift som gir maksimal lineær kompleksitet i forhold til tidligere resultater og eventuelt finne parametre for disse skiftene.

Utførelsen av det praktiske i forbindelse med oppgaven er ikke en stor del av arbeidet. En setter opp ett enkelt skiftregister med tilhørende primitive polynom, velger ønsket starttilstand og genererer maksimalsekvensen. Denne blir så skiftet to ganger og en beregner produktet bitvis $\mathrm{mod}2$. Resultatsekvensen analysereres så ved hjelp av Berlekamp-Massey algoritmen for å finne dens lineære kompleksitet. Dette gjentas for skift av varierende lengde, og en lager tabeller over avvik fra ønskede resultat.

Det som er utfordringen for oss er analyseringen av resultatene. En del av disse vil bli gjengitt helt eller i deler senere, og de vil vise at en har svært mye materiale og

forholde seg til.

1.5.2 Oppbygning

Oppgaven er bygget opp for å stegvis kunne gi en innføring i problemstilling, teori og gjennomføring med påfølgende resultatanalyse. Til slutt en konklusjon over vårt arbeid.

Dette kapitlet har hatt som mål å gi en lett innføring i kryptografi og dens forskjellig reetninger og muligheter. Det har vært vår hensikt med dette å pense inn mot sekvenser spesielt og gi en forståelse for problemstillingen.

Kapittel to tar for seg teorien som underbygger problemstillingen vi har valgt. Denne teorien er ment å gi en pekepinn på hva resultater vi ønsker og finne, i tillegg til å gi et utgangspunkt for eventuell analyse av resultatene. Det er her og vist til hva som allerede finnes av resultater.

Kapittel tre tar for seg hva som er utført, og de umiddelbare resultater. Mulige hypoteser er her lansert.

I kapittel fire vil vi forsøke å forklare og bevise det som er lansert i det foregående kapittel. Forhåpentligvis vil vi her komme med gode teorier og påfølgende bevis av dette. I tillegg vil det her bli presentert noen hypoteser for hva som videre arbeid innenfor dette området kan bringe.

Kapitlet etter inneholder en konklusjon på det arbeid som er nedlagt.

1.5.3 Resultat

De resultatene vi har fått gjennom statistisk analyse av tallmaterialet vårt, og ved hjelp av matematiske bevis, viser at det er muligheter for å kunne garantere større verdier for lineær kompleksitet ved produkter av skift av maksimalsekvenser enn det som tidligere har vært vist og anntatt. Vi mener og at dette kan utvides for flere antall skift.

Kapittel 2

Teoretisk bakgrunn

Dette kapitlet tar for seg den teoretiske bakgrunn og et basisfundament for denne oppgaven. Det blir her presentert noen faktasetninger, teoremer, resultater og en algoritme som alle brukes for å kunne forstå, analysere og beskrive det vi har lagt vekt på og det vi har gjort. En del av dette er elementære byggestener for strømchiffer basert på LFSRer. Andre resultater er setninger som omhandler produkter av maksimalsekvenser som er utgangspunktet for denne forskning. Det vil bli gjengitt resultater som vi håper vi kan bygge videre på. Vi vil også gjengi en alternativ måte å representere en sekvens på. Til slutt blir det presentert et analyseverktøy for beregning av lineær kompleksitet.

2.1 Elementære fakta

I dette avsnittet vil det bli presentert noen definisjoner som er basis for mye av arbeidene innenfor strømchiffer. Disse setningene er av en slik karakter at de blir presentert som faktasetninger. Av samme grunn vil disse heller ikke bli presentert med bevis. For alle setningene antas det at en jobber over $GF[2]$, noe som også er gjeldende for alle påfølgende avsnitt i dette kapittel. Disse utsagnene er en grunnleggende del av teorien som denne tekst og forskning baserer seg på.

Definisjon 2.1 *Et element α er primitivt hvis dets orden er av maksimal verdi, $q^m - 1$. α genererer da alle elementene ulik null i kroppen $GF[2^m]$.*

Definisjon 2.2 *La* f(x) ϵ *GF[2][X] være et polynom av grad minst 1. Da sies* f(x) *være* irredusibelt over GF[2] *hvis det ikke kan skrives som et produkt av to polynomer i GF[2][X], hver av grad større enn 0.*

Definisjon 2.3 *La $f(x)$ være et irredusibelt polynom av grad m. $f(x)$ er et* primitivt *polynom hvis $f(x)\, deler\, x^{2^m-1} - 1$, og $f(x)\, deler\, ikke\, x^r - 1\, for\, 0 < r < 2^m - 1$.*

2.2 Skiftregistre

I denne delen vil det bli presentert en del fakta angående skiftregistre. Vi vil her definere et skiftregister og hvilke egenskaper det må inneha for kunne å generere de ønskede sekvenser av maksimal periode og lengde. Sekvenser på denne form er utgangspunkt for den videre forskningen som vil komme.

Faktum 2.1 Et lineært feedback skiftregister *(LFSR) av lengde* L, *består av L enheter, hver med mulighet for å lagre en bit og som har en utputt- og en innputtlinje.*

Faktum 2.2 *Utputt sekvensen til skiftregisteret blir skrevet:* $s = s_0, s_1, s_2, \ldots\ldots$

Faktum 2.3 *Et LFSR er definert spesielt av et bindingspolynom* $f(x)$*, hvor*

$$f(x) = 1 + c_1 x + c_2 x^2 + \cdots c_L x^L \epsilon GF[2][X] \tag{2.1}$$

Tilstanden til registeret ved tid 0, er definert til å være starttilstanden, som igjen er de første L *elementene i sekvensen.*

$$s_0, s_1, s_2, \ldots, s_{L-1} \tag{2.2}$$

er starttilstanden for et skiftregister av lengde L.

Faktum 2.4 *Sekvensen* s *er definert av* $f(x)$ *og starttilstanden til registeret:*

$$s_j = (c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L}) \bmod 2 for j \geq L \tag{2.3}$$

Resultat sekvensen kan og skrives som en lineær rekursjon på formen

$$s_j + c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L} = 0 for j \geq L \tag{2.4}$$

En omskrivning av (2.4) gir

$$s_j = c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L} for j \geq L \tag{2.5}$$

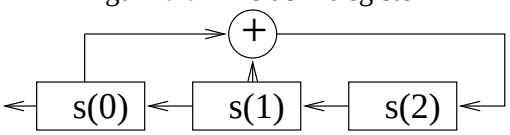
Faktum 2.5 *Hvis* $f(x)$ *er irredusibelt over* $GF[2]$*, vil skiftregisteret generere en sekvens* s *med periode lik det minste tall* N *slik at* $f(x)$ *deler* $1 + x^N$*.*

Faktum 2.6 *Hvis* $f(x)$ *er primitivt vil skiftregisteret generere en sekvens av maksimal periode, en m-sekvens, der perioden er* $2^L - 1$*. Dette vil gjelde for alle starttilstander i registeret bortsett fra* $0 - tilstanden$*.*

Faktum 2.7 *Den* Lineære kompleksiteten *til en endelig sekvens s^n, $L(s^n)$, er definert til å være lengden på det korteste skiftregisteret som kan generere en sekvens som har s^nsom de første n elementer.*

La oss nå se på et eksempel på oppbygning av et skiftregister: $f(x) = 1 + x + x^3$

Figur 2.1: Enkelt skiftregister



Her gir det primitive polynomet $f(x)$ oversikten over hvilke elementer i skiftregisteret som skal adderes sammen $\mod 2$. $f(x)$ er bindingspolynomet, eller generatorpolynomet til den korresponderende sekvensen. Resultatet som etter skiftet settes inn i $s(3)$ blir da beregnet som

$$s(3) = s(0) \oplus s(1)$$

Dette angir da rekursjonligningen ifølge (2.5):

$$s_j = s_{j-3} + s_{j-2}$$

eller

$$s_{j+3} = s_j + s_{j+1}$$

2.2.1 Mer teori

For å kunne regne på hvilken $L(s)$ en gitt sekvens vil ha, må vi innføre et par begreper til.

Vi har hittil sett på Galoiskropper med 2 elementer, $GF[2]$. Vi vil nå se litt nærmere på polynomer og polynomringen $GF[2][X]$. Den inneholder alle polynomer med endelig grad over $GF[2]$. Ved hjelp av irredusible polynomer over $GF[2]$ kan vi lage utvidelseskropper av orden 2^m eller $GF[2^m]$ der m er graden til polynomet. $GF[2^m]$ vil da være en kropp med 2^m elementer. La $p(x)$ være det polynomet som definerer $GF[2^m]$. Alle operasjoner vil da være modulo $p(x)$ i motsetning til modulo p ellers.

Et element $\alpha \epsilon GF[2]$ er en rot i $p(x) \epsilon GF[2][X]$ hvis $p(\alpha) = 0$. Hvis $p(x)$ har grad > 1 er det mulig at den ikke har røtter i $GF[2]$. La da E være en utvidelseskropp av $GF[2]$. $p(x)$ er da sagt å *splitte* over E hvis det eksisterer røtter til $p(x)$ i E. E er sagt å

være en splittkropp til $p(x)$ over $GF[2]$ hvis alle røttene eksisterer der og ingen kropputvidelse av lavere grad inneholder alle disse. Hvis $p(x)$ er irredusibelt over $GF[2]$ vil splittkroppen være $GF[2^m]$ der m er graden til $p(x)$. I tillegg vil alle røttene være gitt på formen $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{m-1}}$. Det er vanlig å kalle disse elementene i $GF[2^m]$ for *konjugerte* av α. Det polynom av minst grad hvor α er rot kalles *minimalpolynomet* til α, $m_\alpha(X)$.

α er sagt å være et *primitivt element* hvis α har orden av maksimal verdi, $2^m - 1$. Da 2 og $2^m - 1$ er relativt primiske har alle de konjugerte røttene til α samme orden og hvis α er et primitivt element vil alle disse også være det og $m_\alpha(X)$ er et primitivt polynom. Når α er et primitivt element kan alle elementer β representeres som en potens av α. Da alle disse elementene, β, har et minimalpolynom, vil det faktum at β tilhører et sett med konjugerte røtter dele opp alle elementene i $GF[2]$ i disjunkte sett. Disse forskjellige settene kalles *syklotomiske kosett.* Det syklotomiske kosettet som inneholder k består av alle distinkte heltall på formen $k2^i$, $i \geq 0$.

2.3 Tracefunksjonen

I dette avsnittet vil vi beskrive et nyttig verktøy som brukes i sekvensrepresentasjon og sekvensanalyse. Tracefunksjonen, med matematisk notasjon Tr, er egentlig en enkel avbildning fra $GF[q^m]$ til $GF[q^n]$, hvor den sistnevnte kroppen er en undermengde av den første:

$$GF[q^m] \supset GF[q^n] \Leftrightarrow n \mid m \tag{2.6}$$

Tracefunksjonen blir da følgende:

$$Tr_n^m(x) : GF[q^m] \longrightarrow GF[q^n] \tag{2.7}$$

Tracefunksjonen fra m til n kan da skrives på formen:

$$Tr_n^m(x) = x + x^{q^n} + x^{q^{2n}} + \cdots + x^{q^{(\frac{m}{n}-1)n}} \tag{2.8}$$

Av disse tre ligningene er det gitt:

Lemma 2.1
Trace avbildningen til x fra $GF[q^m]$ til $GF[q^n]$ er inneholdt i $GF[q^n]$, eller:

$$Tr_n^m(x) \epsilon GF[q^n] \tag{2.9}$$

Bevis:
La $y = Tr_n^m(x)$. Det som da må vises er at $y^{q^n} = y$. Vi har

$$\begin{aligned} y^{q^n} &= (x + x^{q^n} + \cdots + x^{q^{(\frac{m}{n}-1)n}})q^n \\ &= (x^{q^n} + x^{q^{2n}} + \cdots + x^{q^{(\frac{m}{n})n}}) \end{aligned}$$

Dette gir følgende:

$$y^{q^n} = (x^{q^n} + x^{q^{2n}} + \cdots + x^{q^{(\frac{m}{n}-1)n}} + x) \tag{2.10}$$

Ser vi på dette resultatet sammen med (2.9) ser vi at $y^{q^n} = y$.

I resten av avsnittet har vi definert Tracefunksjonen som en avbildning fra $GF[2^n] \longrightarrow GF[2]$.

Lemma 2.2
La $Tr_1^n(x) = x + x^2 + \cdots + x^{2^{n-1}} \epsilon GF[2]$. *Da vil* $Tr_1^n(x)(Tr_1^n(x) + 1) = 0, \forall\, x$.

Lemma 2.3
Tracefunksjonen vil tilfredsstille følgende:

$$Tr_1^n(x) = \begin{cases} 0\, for\, 2^{n-1}\, verdier\, x\, \epsilon\, GF[2^n] \\ 1\, for\, 2^{n-1}\, verdier\, x\, \epsilon\, GF[2^n] \end{cases} \tag{2.11}$$

Bevis:
Resultatene av Tracefunksjonen er jevn fordelt. Det gir følgende:

$Tr_1^n(x) = 0$, har $\leq 2^{n-1}$ løsninger da graden av $Tr_1^n(x) = 0$ er 2^{n-1}
$Tr_1^n(x) = 1$, har $\leq 2^{n-1}$ løsninger da graden av $Tr_1^n(x) = 1$ er 2^{n-1}

Av dette ser vi at likhet må gjelde for begge tilfeller og så bevist.

2.3.1 Eksempel

For å illustrere hva Tracefunksjonen kan hjelpe oss med, vil vi her følge opp figur (2.1) med et enkelt men illustrativt eksempel. Bruker polynomet $f(x) = x^3 + x + 1$. Dette bestemmer da at vi har en avbildning fra $GF[2^3]$ til $GF[2]$. For ligning (2.8) setter vi inn og får

$$Tr_1^3(x) = x + x^{2^1} + x^{2^2}$$

Lar α være en primitiv rot $\Rightarrow\ \alpha^3 = \alpha + 1$. Da vil

$$Tr_1^3(\alpha^n) = \alpha^{1n} + \alpha^{2n} + \alpha^{4n}$$

Dette gir da følgende resultat fra funksjonen:

$$\begin{aligned} Tr(1) = 1^1 + 1^2 + 1^4 &= 1 \\ Tr(\alpha) = \alpha^1 + \alpha^2 + \alpha^4 &= 0 \\ Tr(\alpha^2) = \alpha^2 + \alpha^4 + \alpha^8 &= 0 \\ Tr(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^{12} &= 1 \\ Tr(\alpha^4) = \alpha^4 + \alpha^8 + \alpha^{16} &= 0 \\ Tr(\alpha^5) = \alpha^5 + \alpha^{10} + \alpha^{20} &= 1 \\ Tr(\alpha^6) = \alpha^6 + \alpha^{12} + \alpha^{24} &= 1 \end{aligned}$$

Sekvensen som skiftregisteret i figur (2.1) vil produsere er 1001011 ved starttilstand 100, som er identisk med det resultatet en fikk fra Tracefunksjonen. Ved en forskjellig starttilstand vil sekvensen være en annen, derfor bør en ved bruk av sekvenser og Tracefunksjonen undersøke hvilken starttilstand som er sammenfallende med denne.

2.3.2 Flere resultater

Som beskrevet tidligere er hovedvekten av vårt arbeid lagt ned i produkter av forskjellige skift av samme sekvens og dette reultats kompleksitet. Tracefunksjonen kan hjelpe oss å finne forskjellige sekvenser, og det vi nå skal vise er at en får forskjellige skift av en sekvens ved å se på Tracefunksjonen. La oss starte med utgangssekvensen s_t på Trace form

$$s_t = Tr_1^n(\alpha^t)$$

La så u_t være sekvensen etter τ skift, det vil si

$$u_t = Tr_1^n(\alpha^\tau \alpha^t)$$

En omskrivning av dette gir

$$u_t = Tr_1^n(\alpha^{\tau+t})$$

Det vi vil vise er at ved forskjellige valg av τ så vil sekvensen bli skiftet ulikt i forhold til andre verdier av τ. Vi har to sekvenser u_t og v_t der disse er på formen

$$u_t = Tr_1^n(a\alpha^t),\ v_t = Tr_1^n(b\alpha^t)$$

der a og b kan sees som henholdsvis τ_1 og $\tau_2 \, \epsilon \, GF[2^n]$. Det interessante er å vite når $(u_t) = (v_t)$, eller $u_t = v_t \; \forall \; t$.

$$\begin{aligned} u_t - v_t &= Tr_1^n(a\alpha^t) - Tr_1^n(b\alpha^t) \\ &= Tr_1^n((a-b)\alpha^t) \\ &= Tr_1^n(c\alpha^t) \end{aligned}$$

$u_t = v_t \; \forall \; t$ er det samme som å si $u_t - v_t = 0$. Skal $Tr_1^n(c\alpha^t) = 0 \; \forall \; t$, kan vi se av det foregående at dette kun er tilfelle når $c = 0$, eller at $a = b$. Dette viser at ved forskjellig valg av τ_1 og τ_2 vil vi få forskjellige skift av samme sekvens. Ved å variere variabelen τ, kan en få hvilket skift en ønsker i Traceframstillingen.

2.4 Keys resultat

En annen metode for å beskrive sekvenser er ved bruk av teori om Galois-kropper. Dette er et av emnene Key [4] tar opp i sin artikkel. Han gir videre også et bevis på antall røtter som eksisterer i et produkt av to faser av samme maksimalsekvens. Vi vil her gå gjennom dette beviset. Sammenhengen med vår forskning er det at vi vil utvide hans resultat.

Gitt resultatsekvensen u_t etter et produkt av en sekvens og et skift av denne. Dette gir $u_t = s_t s_{t+\tau}$. Fra forrige avsnitt vet vi at en sekvens s_n kan skrives på formen

$$s_t = \sum_i \alpha^{2^i t}$$

og et skift av denne på formen

$$s_{t+\tau} = \sum_i \alpha^{2^i (t+\tau)}$$

Vi får da for u_t

$$u_t = \sum_{i=0}^{m-1} \alpha^{2^i (t+0)} \sum_{j=0}^{m-1} \alpha^{2^j (t+\tau)} \tag{2.12}$$

Der m er lengden til skiftregisteret. Årsaken for bruken av $t + 0$ er for å enkelt skille faktorene. Trekker vi sammen for å få en ryddig likning får vi

$$u_t = \sum_{i,j} \alpha^{2^{(i+j)} t} \alpha^{2^j \tau}$$

$$= \sum A_e \alpha^{et}$$

Det vi ser nå er at e er et heltall representert på binær form. Analyserer vi dette ser vi at hvis $i = j$ så har vi bare en ikke-null bit, mens hvis $i \neq j$ da har vi 2 ikke-null biter.

$$\begin{aligned} i = j &\Rightarrow e = 2^{i+1} \\ i \neq j &\Rightarrow e = 2^i + 2^j \end{aligned}$$

Når $i = j$ er det ifølge likning (2.12) m tilfeller. I tilfellet med $i \neq j$ er det $m(m-1)/2$ slike heltall. Summert opp er det $m + m(m-1)/2 = m(m+1)/2$ tilfeller.

Det vi da må vise er at A_e ikke blir null i noen tilfeller. Når A_e er på formen $\alpha^{2^k \tau}$ for vilkårlige τ så kan ikke den være lik null.

For det generelle tilfellet kan en da si at antall røtter som kan være tilstede i et skiftregister av lengde m med d sekvenser kombinert med multiplikasjon er

$$_mN_d = \sum_{i=1}^{d} \binom{m}{i} \tag{2.13}$$

2.5 Teoremer

I dette avsnittet presenterer vi to grunnleggende teoremer og deres korresponderende bevis som vi har tatt utgangspunkt i for denne teksten. Teoremene og deres bevis er hentet fra Rueppel[2]. Disse teoremene tar for seg lignende sekvenser som de vi vil presentere i neste kapittel. Senere er det presentert hvilken $L(s^n)$ en får ved å ta produktet av to m-sekvenser hvor sekvens nummer to er et skift av den opprinnelige maksimalsekvens. Det er i disse teoremene, bevist en nedre skranke for $L(s^n)$ hvor en har produktet av flere ekvidistante skift av den samme sekvensen. Produktet det her refereres til er multiplikasjon $\mathrm{mod}2$.

Teorem 2.1 [2]: *Rot eksistens test for produkt av ekvidistante faser av en maksimalsekvens*
La $s_{\tau_1}, s_{\tau_2}, \ldots, s_{\tau_k}$ være k distinkte faser av samme sekvens s, hvis minimalpolynom $m_s(x) \epsilon GF[2][x]$ er primitivt og er av grad L. La $\alpha \epsilon GF[q^L]$ være en rot i $m_s(x)$ og la z være produktet av de k fasene, dvs.:

$$z = s_{\tau_1} s_{\tau_2} \cdots s_{\tau_{k-1}} s_{\tau_k} = \prod_{i=1}^{k} s_{\tau_i} \tag{2.14}$$

La $w_2(j)$ betegne Hammingvekten til den binære representasjonen av et heltall j. Da

er α^e, der $w_2(e) = k$, en rot i minimalpolynomet til z hvis og bare hvis

$$A_e = \begin{vmatrix} \alpha^{\tau_1 2^{e_1}} & \cdots & \cdots & \alpha^{\tau_k 2^{e_1}} \\ \alpha^{\tau_1 2^{e_2}} & \ddots & & \alpha^{\tau_k 2^{e_2}} \\ \vdots & & \ddots & \vdots \\ \alpha^{\tau_1 2^{e_k}} & \cdots & \cdots & \alpha^{\tau_k 2^{e_k}} \end{vmatrix} \neq 0$$

og hvor $e = 2^{e_1} + 2^{e_2} + \cdots + 2^{e_k}$, $0 \leq e_1 < e_2 < \cdots e_k < L$.

***Merknad**:*

Siden røttene alltid er gruppert i sett av konjugerte, holder det å undersøke de eksponentene, e, som er kosettledere. Der finnes $\binom{L}{k}$ distinkte røtter α^ehvor eksponentene har vekt $w_2(e) = k$.

***Bevis**:*

Anta, uten å tape generalitet, at s er i sin karakteristiske fase, dvs

$$s_j = Tr_1^L(\alpha^j) = \alpha^j + \alpha^{2j} + \cdots + \alpha^{2^{L-1}j} \tag{2.15}$$

Ved så å bruke skiftegenskapen til sekvenser med et irredusibelt minimalpolynom, dvs

$$s_{\tau_j} = Tr_1^L(\alpha^t \alpha^{\tau_j}) \tag{2.16}$$

får vi for den j-te biten i z

$$z_j = \prod_{i=1}^{k} Tr_1^L(\alpha^{t_i}\alpha^{\tau_j}) = \prod_{i=1}^{k}(\alpha^{t_i}\alpha^{\tau_j} + \alpha^{2t_i}\alpha^{2\tau_j} + \cdots + \alpha^{2^{L-1}t_i}\alpha^{2^{L-1}\tau_j}) \tag{2.17}$$

(2.17) kan deles opp i to mindre summer, en hvis røtter α^e har vekt $w_2(e) < k$, og en hvis røtter α^e har vekt $w_2(e) = k$. Ved å skrive den første summen som y_j, får vi

$$z_j = y_j + \sum_{\{e:w_2(e)=k\}} A_e \alpha^{e_j} \tag{2.18}$$

hvor A_e betegner den resulterende koeffisienten av α^e. For å kunne få en eksponent e med binær vekt k fra k ledd av toerpotenser, må alle disse leddene være forskjellige. Det vil si $e = 2^{e_1} + 2^{e_2} + \cdots + 2^{e_k}$, hvor $0 \leq e_1 < e_2 < \cdots < e_k < L$. Da det er $\binom{L}{k}$ mulige måter å velge k forskjellige toerpotenser av L mulige, inneholder summen i (2.18), $\binom{L}{k}$ ledd.

En spesifikk eksponent $e = 2^{e_1} + 2^{e_2} + \cdots + 2^{e_k}$, $0 \leq e_1 < e_2 < \cdots < e_k < L$ kan en finne på $k!$ forskjellige måter som korresponderer til de $k!$ mulige permutasjonene av de k forskjellige toerpotensleddene. Med andre ord er det $k!$ valg for de distinkte $m_1, \ldots, m_k$ slik at $e = 2^{m_1} + 2^{m_2} + \cdots + 2^{m_k}$. Dette korresponderer til antall mu-

lige måter en kan velge de k forskjellige toerpotensene, hver fra en forskjellig Trace i (2.17). Den resulterende koeffisienten til α^{ej} i (2.18) vil bli

$$A_e = \sum_{\overline{m}\epsilon P_e} (\alpha^{\tau_1})^{2^{m_1}} (\alpha^{\tau_2})^{2^{m_2}} \cdots (\alpha^{\tau_k})^{2^{m_k}} \tag{2.19}$$

hvor P_e er settet av alle permutasjoner av $(e_1, \ldots, e_k)$.

Tar en til etterretning at addisjon og subtraksjon er identisk over enhver kropp med karakteristikk 2, følger det at (2.19) beregner $k \times k$ determinanten definert i (2.15). Vi konkluderer med at bare når denne determinanten er ulik 0 for en spesifikk eksponent e, vil roten α^e bidra til $L(z)$.

Teorem 2.2 [2]: *Produkt av ekvidistante faser: Når $s_t, s_{t+\tau}, ..., s_{t+(k-1)\tau}$, $1 \leq k \leq L$, er ekvidistante faser av samme maksimalsekvens s og $\gcd(\tau, 2^L - 1) = 1$, da er alle $\binom{L}{k}$ distinkte elementer $\alpha^e \epsilon GF[2^L]$, hvis eksponenter e har binær vekt $w_2(e) = k$, garantert å være røtter i minimalpolynomet til produktet av alle de k distinkte fasene, $m_z(x)$. Lineær kompleksiteten til produktsekvensen z er da begrenset nedad til $L(z) \geq \binom{L}{k}$*

Bevis: *Anta uten ved tap av generalitet at $t, t+\tau, \ldots, t+(k-1)\tau = 0, \tau, \ldots, (k-1)\tau$, det vil si*

$$z = s * s_\tau * \cdots * s_{(k-1)\tau} \tag{2.20}$$

Da vil koeffisient-determinanten som definert i Teorem (2.1) bli en Vandermonde-determinant ΔV hvis verdi er gitt ved

$$A_e = \Delta V(\alpha^{\tau 2^{e_1}}, \ldots, \alpha^{\tau 2^{e_k}}) = \prod_{n=2}^{k} \prod_{j=1}^{n-1} (\alpha^{\tau 2^{e_n}} - \alpha^{\tau 2^{e_j}}) \tag{2.21}$$

Vandermonde-determinanten er null hvis og bare hvis $\tau 2^{e_n} = \tau 2^{e_j}$ for minst ett par $n \neq j$. Men dette kan ikke skje da e har en unik oppbygning $2^{e_1} + \cdots + 2^{e_k}$ med $0 \leq e_1 < e_2 \cdots < e_k < L$, og siden, når $\gcd(\tau, 2^L - 1) = 1$, vil multiplisering med τ resultere i en permutasjon av alle elementene i ringen av heltall $\mod 2^L - 1$. Derfor er alle de $\binom{L}{k}$ distinkte elementene $\alpha^e \epsilon GF(2^L)$ røtter til $m_z(x)$ og derfor vil bidra til $L(z)$.

2.5.1 Eksempel

For å illustrere teorien vi nå har gjennomgått skal vi lage et enkelt eksempel. Vi vil finne de røttene α^e som ikke forsvinner, det vil si de som medvirker i verdien til $L(s)$.

Da disse røttene er gruppert i sett av konjugerte trenger vi bare å undersøke de røttene der verdien til eksponenten til α^e er kosettleder.

Anta at maksimalsekvensen vår er generert ved hjelp av det primitive polynomet $x^4 + x^3 + 1$ over $GF[2]$. Traceframstillingen til bit j er

$$s_j = Tr_1^4(\alpha^j) = \alpha^j + \alpha^{2j} + \alpha^{4j} + \alpha^{8j}$$

La oss se på produktet av to faser av samme sekvens, det vil si $z = s_t s_{t+\tau}$. La $\tau = 2$. I $GF[2^4]$ er det kun α^3 og α^5 som har eksponenter som er kosettledere som igjen har binær vekt lik 2. Setter opp matrisen for A_e som beskrevet i forrige avsnitt. For $e = 3$ har vi $e_1 = 0$, $e_2 = 1$ dette da $2^0 + 2^1 = 3$. $t_1 = 0$, $t_2 = t_{1+\tau} = 2$.

$$A_3 = \begin{vmatrix} 1 & \alpha^2 \\ 1 & \alpha^4 \end{vmatrix} = \alpha^4 + \alpha^2 \neq 0$$

Gjør det samme for $e = 5$

$$A_5 = \begin{vmatrix} 1 & \alpha^2 \\ 1 & \alpha^8 \end{vmatrix} = \alpha^8 + \alpha^2 \neq 0$$

Dette viser at minimalpolynomene til α^3 og α^5 begge er røtter i generatorpolynomet til z. Det vil si at røttene til disse minimalpolynomene er røtter i dette. Ser vi på kosettene $\{3, 6, 12, 9\}$, $\{5, 10\}$ og teller opp disse ser vi at den nye sekvensen har minst 6 røtter.

Vi må nå regne ut for $w(e) = 1$. I avsnittet hvor vi presenterte Keys resultat (2.4), fant vi at hvis $i = j$ så vil $e = 2^{i+1}$. Dette gjelder når $w(e) = 1$. Antallet slike tall ble vist til å være lik m, som igjen er graden til generatorpolynomet. I dette eksempelet er den lik 4. Da får vi totalt antall røtter til å være lik 10, og

$$\sum_{i=1}^{2} \binom{m}{i} = 4 + 6 = 10$$

2.6 Berlekamp-Massey

For å kunne beregne den lineære kompleksiteten til en endelig sekvens s^n, har vi tatt utgangspunkt i en enkel algoritme. Denne algoritmen er utformet av Berlekamp og Massey[5]. Den baserer seg på bruk av rekusive matematiske ligninger. Denne algoritmen brukes også innenfor kodeteori til å dekode BCH koder.

Algoritmen tar inn, i vårt tilfelle, en endelig binær sekvens og dens korresponderende lengde. Ved å bruke, på en sekvens av lengde N, $O(N^2)$ bitoperasjoner, returnerer den lengden på det korteste skiftregisteret som kan generere sekvensen, det vil si $L(s^N)$. Med en liten endring kan den også returnere bindingspolynomet til skiftre-

gisteret. Algoritmen trenger ikke mer enn $2L$ bits fra et skiftregister av lengde L for å kunne returnere disse to variablene. Noe som tidligere nevnt, er en uheldig svakhet til et enkelt skiftregister.

En liten forklaring på hva denne algoritmen egentlig gjør er på sin plass. Den er egentlig en enkel rekursjonsligning. La oss se på rekursjonen fra $L(s^{n-1})$ til $L(s^n)$. Forskjellen mellom den n-te biten, s_{n-1}, og den n-te biten som blir generert av LFSRen som genererer s^{n-1}, kalles den neste diskrepansen δ_{n-1}. Hvis denne LFSRen klarer å generere denne biten slik at forskjellen er lik null, vil den lineære kompleksiteten ikke forandre seg, men er det en forskjell og den eksisterende verdien for denne er mindre enn $\frac{n}{2}$, så vil den oppdateres. Tallet $\frac{n}{2}$ stammer fra det faktum at den trenger $2n$ bits for å finne $L(s^n)$.

De to følgende ligningene er basisen for algoritmen.

$$\begin{aligned}
\delta_{n-1} = 0 \quad &\Rightarrow \quad L(s^n) = L(s^{n-1}) \\
\delta_{n-1} = 1 \quad &\Rightarrow \quad L(s^n) = L(s^{n-1})\, hvis\, L(s^{n-1}) > \frac{n}{2} \\
&\qquad\quad L(s^n) = n - L(s^{n-1})\, hvis\, L(s^{n-1}) \leq \frac{n}{2}
\end{aligned}$$

Algoritmen blir her presentert med pseudokode:

Algoritme 1 Berlekamp-Massey

$Initialisering : C(D) = 1,\ L = 0,\ x = 1, C^*(D) = 1,\ n = 0$
$Input : s^N = Sekvensen,\ N = Lengden$
$while\ n < N\ do$
  /*Beregn den neste diskrepansen*/
  $\delta = s_n + c_1 s_{n-1} + ... + c_L s_{n-L}$
  if $\delta = 0$
    $x = x + 1$
  else
    $T(D) = C(D)$
    $C(D) = C(D) + *D^x * C^*(D)$
    if $2L \leq n$
      $L = n + 1 - L$
      $x = 1$
      $C^*(D) = T(D)$
    else
      $x = x + 1$
  $n = n + 1$
return $L, C(D)$

Kapittel 3

Gjennomføring

Bakgrunnen for at vi valgte å se nærmere på sekvenser som et produkt av tre vilkårlige, ulike skift av samme sekvens, var at det ikke er mye publisert forskning på dette området, annet enn for to slike skift av samme maksimalsekvens. Vi ville prøve å finne en sammenheng mellom hvilke skift av disse sekvensene som gav en maksimal lineær kompleksitet, eller om om noen av sekvensene egner seg bedre til bruk ann andre. Dette kan gi pekepinner på hvilke som kan brukes som byggestener for forskjellige applikasjoner innenfor kodeteori og kryptografi. En har visse ideer om hva som er mulig å brukes. Dette inneholder for eksempel at bindingspolynomet, eller generatorpolynomet, har generelt bedre egenskaper når graden er primitiv.

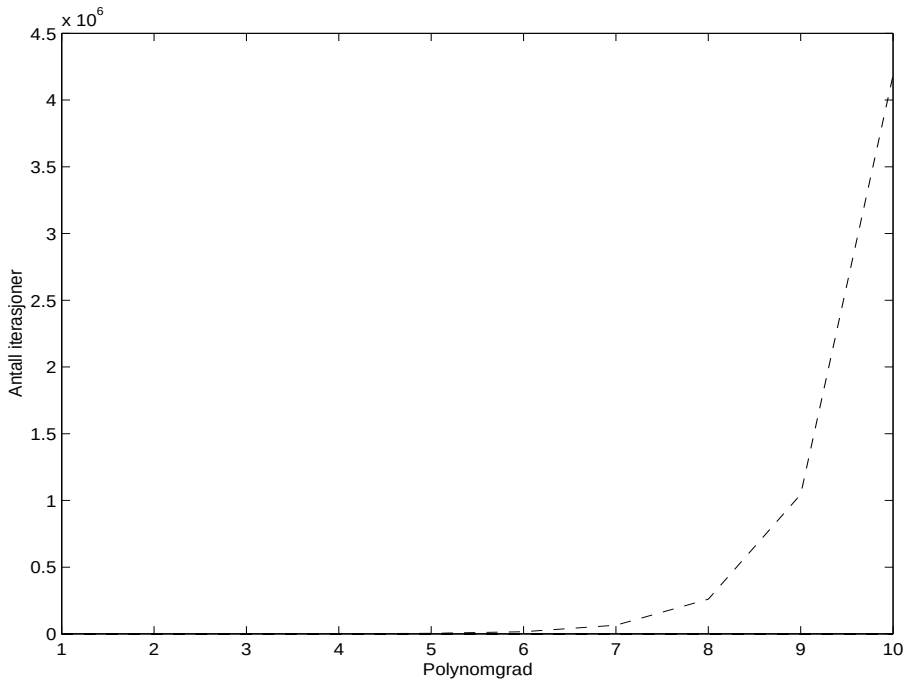
Utover det som er presentert av Key[4] og Rueppel[2] er det ikke mye arbeid gjort på dette området og mulighetene for å kunne finne nye mønstre er der. Disse to punktene gjør arbeidet vi har startet meget interessant.

3.1 Om oppbyggingen

Selve delen hvor programmering er inne i bildet, det vil si den praktiske delen med genereringen av sekvensene, multipliseringen $\mathrm{mod}2$ og testing for kompleksitet er forsåvidt enkel og kan sees på som elementær programmering og ikke en stor del av arbeidet. Vi konstruerte et skiftregister som kan ta som innparametre, et polynom av hvilken grad en måtte ønske og den ønskede starttilstand for skiftregisteret. Dette gjør det mulig å generere hvilken som helst sekvens innenfor de forutsatte parametre. I tillegg gjør muligheten for å velge starttilstand det mulig å kontrollere de forskjellige sekvensene ved bruk av Tracefunksjonen som er beskrevet i avsnitt (2.3).

Ettersom Berlekamp-Massey algoritmen (2.6) er en algoritme av orden $O(n^2)$, der n er lengden av bitsekvensen som skal undersøkes, er det i vårt spesielle tilfelle en begrensning i muligheter for valg av grad til polynomene. Kjøretiden for programmet vil være svært lang hvis en vil finne mønstre for polynomer av grad > 13. Dette er

illustrert i figur (3.1) under. Den viser hvordan antall beregninger vokser i forhold til graden av generatorpolynomet. Bakgrunnstallene for denne figuren er listet i (Tillegg B).



Figur 3.1: Kjøretid

Det vi likevel mener å oppnå med våre tester er at vi skal kunne finne et mønster i data fra polynomer av mindre grad, uten å tape generalitet, for så å kontrollere dette med resultatene fra de av høyere grad ved å generere tilfeldige bruddstykker av dem.

De primitive polynomene som vi har brukt, noe som er essensielt for at det her skal kunne bli generert et akseptabelt resultat, er listet opp i (Tillegg B). Det er her også gitt starttilstandene til de forskjellige skiftregistrene. Disse starttilstandene er viktige når en skal kontrollere korrektheten av generatoren ved hjelp av Tracefunksjonen. En kontroll som må utføres for å kunne bekrefte resultatene.

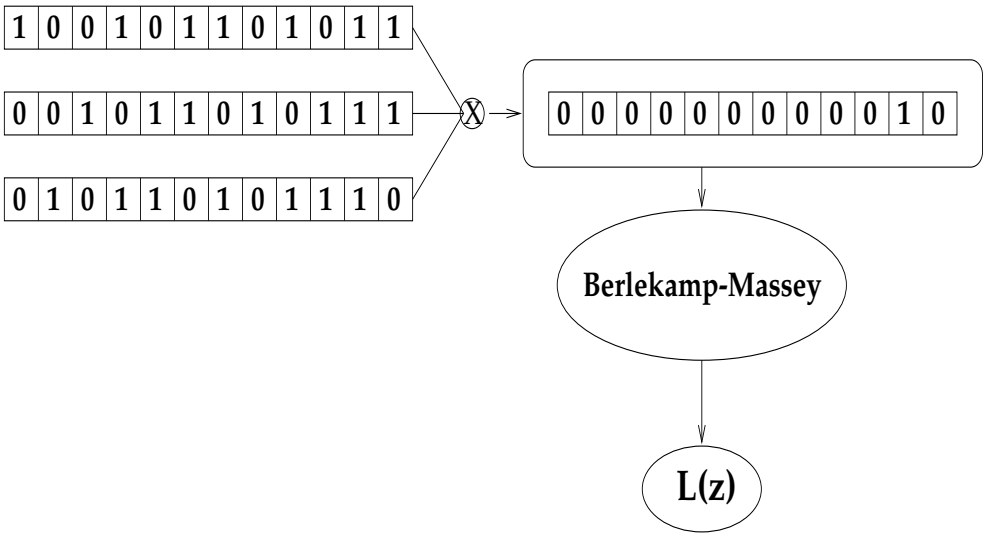
Programmet fortsetter etter genereringen av sekvensene med en beregning av den lineære kompleksiteten. Produktsekvensen, eller resultatsekvensen, blir sendt til Berlekamp-Massey algoritmen, som igjen gir som resultat den lineære kompleksiteten til sekvensen. Dersom denne er lavere enn forventet, lar vi programmet skrive det og hvilket skift av originalsekvensen vi har beregnet for til fil, slik at det er mulig å holde kontroll på resultatene.

Den forventete verdi som her er nevnt, er beregnet utfra teorien som er beskrevet i (2.4). Dette vil si at vi håper å få flest mulig resultater på formen

$$_mN_d = \sum_{i=1}^{d} \binom{m}{i}$$

For mindre sekvenser blir det også tilfeldig kontrollert at de korrekte sekvensene er generert ved hjelp av teorigrunnlaget i (2.3). Dette gir oss grunnlag for å kunne forvente at sekvensene av større format er korrekte med hensyn til både generering, lengde og kompleksitet. Dette er viktig i det henseende at det arbeid som er gjort kan brukes på en vitenskapelig måte, og for senere å kunne verifisere teorier og hypoteser.

Fremgangsmåten som er beskrevet i det forgående avsnitt kan sees på følgende måte som i figur (3.2) under.



Figur 3.2: Programmet

3.2 Behandling av data

Programdelen i forskningen har som hovedformål å presentere kompleksiteten til resultatsekvensen. Programmet er designet og utformet på en slik måte at utskriftene angir kompleksiteten til produktsekvensen av de tre skiftene, og nødvendigvis også hvilke skift som har definert grunnlaget for sekvensen. Dette faller naturlig. Resultatene er skrevet til rene tekstfiler hvor ingenting annet enn resultater og deres korresponderende skift blir presentert. For polynomer av grad > 10 blir det desverre svært store filer, selv om en ikke skriver ut mere enn avvikene fra forventet beste resultat. Dette gjør at vi må prøve å finne sammenhenger i resultatene fra de mindre filene, det vil si filer korresponderende til polynomer av grad < 10. Det som da senere vil bli

gjort er å prøve ut hypotesene som en kan komme til på de resterende filene med resultater og muligens bekrefte disse. Dette er med på å gjøre at en kan få et repektabelt resultat selv om en ikke har gått gjennom hver linje i hver fil. Hvis vi ikke finner sammenhenger og mulige hypoteser i de mindre filene er det kanskje nødvendig med en grundig inspeksjon av filene korresponderende til polynomene av større grad, men det er i stor grad usannsynlig at det skal kunne finnes noe der som en ikke har sett spor av tidligere. Tabeller med resultater er listet i (Tillegg B).

3.3 Resultater

Når vi etterhvert fikk inn litt resultater kunne vi fort konstatere at det var forskjell på polynomer hvis grad var av primtalls form og de som var på sammensatt form. Ofte var det meget store forskjeller i resultater. Dette gjør at vi har valgt å presentere resultatene i to deler.

Som en merknad kan det nevnes at når det her snakkes om reduksjon i lineær kompleksitet, så er det i forhold til ligningen

$$\sum_{i=1}^{3}\binom{m}{i}$$

der m er graden til generatorpolynomet. Denne ligningen er et spesialtilfelle av den som er presentert i avsnitt (3.1).

3.3.1 Polynomer med sammensatt grad

Det umiddelbare resultat som kan trekkes ut av alle våre forsøk, er det faktum at binære sekvenser generert av polynomer hvis grad er på ikke-primtalls form, så vil den lineære kompleksiteten være meget ustabil. Nærmere halvparten av alle resultatproduktene viser seg å ikke generere fullstendig lineær kompleksitet som nevnt i tidligere avsnitt. En av de umiddelbare forklaringene på dette kan være at dette er forårsaket av at faktorene i polynomet $x^{2^m-1}-1$ virker inn på resultatet. For å kunne se dette, la oss se nærmere på sekvenser generert av et polynom med grad fire og med følgende faktorisering:

$$(x^{2^4-1}-1) \bmod 2 = (x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)$$

Her ser vi at $x^{15}-1$ består av en del elementer av lik- og forskjellig grad. Ved nærmere undersøkelse av resultatfilen for det valgte primitive polynomet, (x^4+x+1), finner en at kompleksiteten varierer mellom 10 og 12 bortsett fra 0. Ser en dette i sammenheng med graden til de forskjellige elementene av $x^{15}-1$, så kan en muligens

komme til en foreløpig konklusjon om at disse elementene reduserer kompleksiteten med en totalverdi tilsvarende graden deres. Det som er interessant er at en får og en innvirkning fra produktene av disse elementene. Disse produktene er, ved en overfladisk gjennomgang, ikke av en slik form at flere av elementene med maksimal grad ikke er representert ved multiplisering. Det er kun et element av maksimal grad med i produktene. Skal vi følge eksempelet her så vil det si at kun ett av polynomene av grad 4 som er faktor i $x^{15} - 1$ opptrer samtidig. Noe som kan indikere at en aldri, untatt null, vil få en reduksjon på større eller lik $2m$ i polynomer på denne form, der m angir graden tl generatorpolynomet. Ved videre å undersøke resultater generert av andre polynomer på samme form, har vi funnet at dette kan bekreftes også for disse.

En kan og sammenligne disse resultatene med teorien til Rueppel (2.5), der denne reduksjonen i kompleksitet kan forklares, eventuelt beskrives, ved å se på minimalpolynomets røtter og deres vekt. En kan da finne, ved regning, hvilke røtter som kanselleres.

Et generellt problem angående polynomer på denne form, er at de gir generellt sett dårlige resultater og opp imot 50% av skiftene gir redusert kompleksitet. Dette er med på å bekrefte det som Key nevnte i ([4]).

3.3.2 Polynomer med primtalls grad

Kunne en ikke bruke polynomene definert og presentert i forrige avsnitt til så mye annet enn forskning, er resultatene for polynomene vi testet for som hadde grad på primtalls form mer oppløftende. Her er det en reduksjon i kompleksitet, som for de foregående, men de reduseres ikke med et antall som tilsvarer produktet av disse. Reduksjonen skjer for mye færre antall av skiftene og reduksjonen er mindre og mer forutsigbart. Det er denne forutsigbarheten som gjør resultatene så oppløftende.

La oss se her på sekvenser generert av et polynom av grad syv:

$$(x^{2^7-1} - 1) \bmod 2 = (x + 1)(x^7 + x + 1) \cdots$$

$x^{127} - 1$ faktoriserer i et førstegrads polynom og flere syvendegrads polynomer. Resultatene for sekvenser generert av $x^7 + x + 1$ viser at reduksjonen i kompleksitet for de forskjellige produktene går ned til 56 eller 0 og kun disse to verdiene. Dette kan i utgangspunktet være med å gi en pekepinn på at en ikke får en reduksjon tilsvarende produktet av potenser, men bare for ett av elementene i faktoriseringen. En mulig måte å angripe problemet på da er å undersøke om det er et mønster i hvilke polynomer som bortfaller. Videre resultater viser at tilsvarende gjelder også for resten av testpolynomene.

Den eneste reduksjonen som det kan vise seg at en vil få her, også unntatt null,

vil være på m eller tilsvarende graden på generatorpolynomet. Det som også kan vise seg å være av interesse er at antall resultater som gir verdien 0 er i stort mindretall. En forklaring på dette fenomenet kan være at verdien av $\binom{r}{i}\, for\, i = 1$ kanselleres. Spørsmålet en her kan stille og som vil være viktig, er ved hvilke skift det viser seg at de røtter α^e med $w(e) = 1$, som beskrevet i avsnitt (2.5), vil forsvinne. Kan en finne mønster for dette er det mulig å beskrive en setning som kan garantere en nedre grense på $L(s)$ til å være

$$\sum_{i=2}^{3}\binom{m}{i}$$

De begynnende analyser av resultatene for polynomer av primtalls grad, viser det seg at for de sekvenser som er generert av tre andre sekvenser hvor der er ekvidistante skift, det vil si produkter på formen

$$z = s_t s_{t+\tau_1} s_{t+\tau_2},\ hvor\, \tau_2 = 2\tau_1$$

vil dette være garantert. En skal altså kunne garantere for noen tilfeller av maksimal lineær kompleksitet. En nærmere analyse av dette tilfellet vil komme i neste kapittel. I tillegg er det også mulig å se andre tilfeller som garantere dette, spesielt hvor $\tau_2 = 2^n\tau_1, n > 1$.

Generelt sett kan en si at denne forskjellen for polynomer av sammensatt grad og de med primtalls grad kan gi en pekepinn på enkle metoder for å sile ut uønskede resultater. Kan en finne metoder for ihvertfall å unngå den totale reduksjonen, vil en kunne garantere en lineær kompleksitet på minst.

$$\sum_{i=1}^{3}\binom{m}{i} - m$$

eller

$$\sum_{i=2}^{3}\binom{m}{i}$$

Dette kan da gi oss en mulighet å generere gode sekvenser på en enkel og billig måte. Dette vil vi se nærmere på i neste kapittel.

Det numeriske beviste for disse hypotesene er gitt i (Tillegg 2). Der er tabellene listet opp med avvikene fra utgangsformelen nevnt tidligere i dette kapitlet.

Kapittel 4

Resultatanalyse

I dette kapitlet vil vi prøve å analysere resultatene som prgrammet beskrevet i forrige kapittel gav oss. Vi vil gå inn i resultatene og om mulig forklare hvorfor kompleksiteten reduseres litt og hvorfor den i enkelte tilfeller reduseres til null. I dette kapittelet vil vi konsentrere oss om de resultatene som er av interesse, det vil si der graden til generatorpolynomet er på primtallsform.

4.1 Maksimal reduksjon

Noe som er meget interessant og som ihvertfall kan garantere en minimal reduksjon av kompleksitet er å beskrive tilfellene der kompleksiteten er lik null, og om mulig gi et lemma på hvilke valg man ikke bør gjøre i skiftparametrene.

Det interessante i denne delen av analysen er å kunne begrunne hvorfor produktet av de opprinnelige sekvensene resulterer i null sekvensen, og beskrive de valg som vil gi dette uønskede resultat. Fremgangsmåten er å se på rekursjonen i genereringen og undersøke om noen av leddene der kanselleres. Hvis alle disse leddene faller bort, vil et gi oss en god forklaring på når kompleksiteten blir lik null. Hvis en kan ved hjelp av dette være istand til å beregne hvilke skift som gir null-sekvensen som resultatsekvens, har en garantert for en selv en nedre grense på lineær kompleksitet.

La oss se på et enkelt, men beskrivende eksempel for sekvenser generert av det primitive polynomet $x^5 + x^2 + 1$. Ved hjelp av definisjonenene i avsitt (2.2) får vi følgende rekursjon

$$s_{t+5} = s_{t+2} + s_t \tag{4.1}$$

La oss så se på sekvensen z generert av

7

$$s_t s_{t+1} s_{t+18} \tag{4.2}$$

som etter Berlekamp-Massey algoritmen viser seg å ha kompleksitet lik null. For at dette skal stemme må s være null-sekvensen. For å finne hva s_{t+18} kan erstattes med bruker vi det primitive elementet α som genererer alle elementene i kroppen $GF[2][x]/(x^{32}-1)$ der x^5+x^2+1 er generatorpolynom. Det vil si at $\alpha^5 = \alpha^2+1$. Ved videre regning gir dette at $\alpha^{18} = \alpha+1$. Det forteller oss at $s_{t+18} = s_{t+1}+s_t$. For så å bruke dette i ligning (4.2) får en følgende likning:

$$\begin{aligned} & s_t s_{t+1}(s_{t+1}+s_t) && (4.3)\\ = \;& s_t^2 s_{t+1} + s_t s_{t+1}^2 && (4.4)\\ = \;& 2 s_t s_{t+1} && (4.5) \end{aligned}$$

(4.5) følger av egenskaper ved regning over $GF[2]$ og resulterer i nullsekvensen. Ved videre å beregne dette produktet for de resterende skiftene i forbindelse med dette generatorpolynomet, får vi som resultat at de gir samme løsning. Ved inngående regning på andre resultater på samme form, får vi at dette gjentar seg. Det er innlysende at noen skift vil kunne nulle ut andre skift og en vil sitte igjen med dette resultatet.

Det vi da hevder er at en kan kontrollere når en får en lineær kompleksitet lik null. Det vil si at en kan luke ut disse tilfellene før de vil forekomme og da være garantert en høy kompleksitetsverdi

La oss da se på det generelle tilfellet,

$$s_t s_{t+\tau_1} s_{t+\tau_2} = 0$$

Dette vil skje når

$$s_{t+\tau_2} = s_t + s_{t+\tau_1}$$

Når er det så dette vil intreffe. Vi har at $f(x)$ er minimalpolynomet til α, og at $f(\alpha) = 0$. I tillegg vet vi at $s_t = Tr(a\alpha^t)$. Dette gir oss da

$$f(x) \,|\, x^{\tau_2} + x^{\tau_1} + 1$$

og

$$\begin{aligned} f(\alpha) &= 0 \\ &\Downarrow \\ \alpha^{\tau_2} + \alpha^{\tau_1} + 1 &= 0 \end{aligned}$$

$$\Downarrow$$

$$s_{t+\tau_2} s_{t+\tau_1} s_t = 0$$

Løsningen på dette blir da at en må velge generellt τ_1 og τ_2 slik at følgende ligning blir oppfylt:

$$Tr[a\alpha^t(\alpha^{\tau_2} + \alpha^{\tau_1} + 1)] = 0$$

Ved generering av slike sekvenser er det mulig å legge dette inn som kontrollsjekk for å unngå forrige resultat, eller en kan velge å kun bruke verdier for τ_1 og τ_2 slik at

$$0 < \tau_1, \tau_2 < m$$

Dette fordi en da unngår en lineær avhengighet mellom skiftene som gir denne reduksjonen.

Ved å følge disse retningslinjene vil en være garantert å unngå null-sekvensen ved en slik generering, og en kan få garantert gode egenskaper.

4.2 Ekvidistante skift

Som beskrevet i (3.3.2), kunne vi ved en grei analyse av resultatene våre observere at ved tilfeller av ekvidistante skift så ville kompleksiteten ikke bli redusert i forhold til vår utvidelse av Keys resultat (2.4), det vil si

$$L(z) = \sum_{i=1}^{3} \binom{m}{i} \tag{4.6}$$

der m er graden til generatorpolynomet. Dette ville innebære en forbedring i forhold til teorien i (2.5) med noen begrensninger. Disse begrensningene er som følger:

- Generatorpolynomet til sekvensene må ha primtalls-grad
- Det er produkt av tre skift av samme sekvens
- De tre skiftene må være ekvidistante

Det vi da må kunne vise og som vil være nytt i denne sammenheng er at ikke bare elementene $\alpha^e, \, der \, w(e) = 3$ ikke vil kanselleres, men at også elementene $\alpha^e \, der \, w(e) = 1, 2$ ikke vil kanselleres. For de med $w(e) = 3$ er dette bevist i teorigrunnlaget.. Måten å vise dette på er å undersøke om A_e i kapittel (2.5) er ulik null for begge disse tilfellene. I så fall vil dette gi oss det ønskede resultat.

4.2.1 Tilfelle med vekt lik to

La oss først se på tilfellet der den binære vekten til e, $w(e) = 2$. Dette gir følgende utgangsrelasjoner

$$w(e) = 2 \Rightarrow e = 2^{e_1} + 2^{e_2} = 2^a + 2^b$$

Dette er basisen som hentet fra (2.5). Vi har multiplisert sammen tre forskjellige skift av samme sekvens, der det ene skiftet er originalsekvensen selv.

La u_t være resultatsekvensen etter multiplikasjonen, dvs.

$$\begin{aligned} u_t &= s_t s_{t+\tau} s_{t+\tau_2} \\ &= \sum_i \alpha^{2^i(t+0)} \sum_j \alpha^{2^j(t+\tau_1)} \sum_k \alpha^{2^k(t+\tau_2)} \\ &= \sum_{i,j,k} \alpha^{2^j\tau_1 + 2^k\tau_2} \alpha^{t(2^i+2^j+2^k)} \\ &= \sum A_e \alpha^{et} \end{aligned}$$

Årsaken til at en får $(t + 0)$ i i-leddet er at s_t er originalsekvensen og ikke skiftet. Det vi interesserer oss for her er verdien til A_e, og at den ikke skal bli lik null.

Dette gir oss

$$\begin{aligned} 2^i + 2^j + 2^k &= e \\ &= 2^a + 2^b \end{aligned}$$

hvor

$$0 \leq a, b < m$$

Dette gir oss følgende muligheter for valg av i, j *og* k.

i	j	k
$a-1$	$a-1$	b
$a-1$	b	$a-1$
b	$a-1$	$a-1$
$b-1$	$b-1$	a
$b-1$	a	$b-1$
a	$b-1$	$b-1$

Den ene av våre forutsetninger sier at det her skal være snakk om ekvidistante skift. Dette vil si at $\tau_2 = 2\tau_1 \; der \; \tau_1 = \tau$. Vi har følgende startligning for A_e

$$A_e = \sum_{j,k} \alpha^{\tau_1 2^j} \alpha^{\tau_2 2^k}$$

der variabelen i som da bortfaller. En helt generell ligning finner man i avsnitt (2.5).

Dette gir oss da

$$\begin{aligned} A_e &= \alpha^{2^{a-1}\tau + 2^b 2\tau} + \alpha^{2^b \tau + 2^{a-1} 2\tau} \\ &+ \alpha^{2^{a-1}\tau + 2^{a-1} 2\tau} + \alpha^{2^{b-1}\tau + 2^a 2\tau} \\ &+ \alpha^{2^a \tau + 2^{b-1} 2\tau} + \alpha^{2^{b-1}\tau + 2^{b-1} 2\tau} \end{aligned}$$

I ligningen over ser vi at vi har to eksemplarer av $\alpha^{2^a\tau + 2^b\tau}$. Disse faller bort, og ved omskrivning får vi

$$\begin{aligned} A_e &= \alpha^{(2^{a-1} + 2^{b+1})\tau} + \alpha^{(2^{a-1} + 2^a)\tau} + \alpha^{(2^{b-1} + 2^{a+1})\tau} + \alpha^{(2^{b-1} + 2^b)\tau} \\ &= (\alpha^{(1 + 2^{b-a+2})\tau} + \alpha^{3\tau} + \alpha^{(2^{b-a} + 4)\tau} + \alpha^{3 * 2^{b-a}\tau})^{2^{a-1}} \end{aligned}$$

For å få et litt enklere uttrykk erstatter vi α^τ med β og $b - a$ med r og får

$$\begin{aligned} A_e &= (\beta^{1+2^{r+2}} + \beta^3 + \beta^{2^r+4} + \beta^{3*2^r})^{2^{a-1}} \\ &= (\beta^3)^{2^{a-1}} (\beta^{2^{r+2}-2} + 1 + \beta^{2^r+1} + \beta^{3*(2^r-1)})^{2^{a-1}} \\ &= (\beta^3)^{2^{a-1}} (\beta^{3*(2^r-1)} + 1)(\beta^{2^r+1} + 1) \\ &\neq 0^? \end{aligned}$$

Det vi da sitter igjen med er å undersøke om dette siste uttrykket kan bli lik null for noen tilfeller.

La oss så se på det første leddet, $(\beta^3)^{2^{a-1}}$. Dette kan ikke bli null med mindre β er lik null. β er igjen lik α^τ, noe som ikke kan bli null. For de to resterende leddene må vi undersøke om et de følgende kan gjelde

$$\begin{aligned} \beta^{3(2^r-1)} &= 1^? \\ \beta^{2^r+1} &= 1^? \end{aligned}$$

Ved å sette inn for β får vi

$$\left.\begin{array}{rcl} \alpha^{3(2^r-1)*\tau} & = & 1^? \\ \alpha^{(2^r+1)*\tau} & = & 1^? \end{array}\right\} \tag{4.7}$$

Generelt er generatorpolynomene vi har brukt ved generering av sekvensene primitive polynomer med vilkårlig grad m. Dette gir en sekvenslengde, ettersom det her er snakk om maksimalsekvenser, på $2^m - 1$. For at da ligningene i (4.7) skal kunne tilfredsstille likhet, må 1 av disse to likningene under gjelde.

$$\begin{array}{rcl} 3(2^r-1)\tau \equiv 0 \mod (2^m-1) & \Rightarrow & \tau \equiv 0 \mod (\dfrac{2^m-1}{\gcd(3(2^r-1),(2^m-1))}) \\ (2^r+1)\tau \equiv 0 \mod (2^m-1) & \Rightarrow & \tau \equiv 0 \mod (\dfrac{2^m-1}{\gcd((2^m-1),(2^r-1))}) \end{array}$$

Det en da må sikre er at modulusen vil være $2^m - 1$, det vil si at $2^m - 1$ er primtall. Da må τ være på formen

$$\tau = n * (2^m - 1)$$

Noe som igjen ikke er mulig da vi da vil sitte igjen med vår opprinnelige sekvens. Dette fordi skiftene har gått rundt n ganger og tilbake til utgangspunktet. Vi får da at sekvensen ikke er forandret i forhold til utgangspunket og produktsekvensen vår er da ikke en gyldig sekvens. Problemet da er at vi må vite hvilken grad polynomet kan ha for at dette skal gjelde. Først og fremst må genaratorpolynomet ha primtalls-grad, men dette er ikke nok. En må i tillegg sikre at $2^m - 1$ er et primtall. For hvilke verdier av m dette vil gjelde for er listet i (Tillegg B).

4.2.2 Tilfelle med vekt lik en

Nå gir vi tilfellet der $w(e) = 1$ vår oppmerksomhet. De tilsvarende relasjoner som vi bygget opp i forrige avsnitt er

$$\begin{array}{rcl} w(e) = 1 \Rightarrow e & = & 2^s \\ 2^i + 2^j + 2^k & = & e \\ & = & 2^s \end{array}$$

hvor

$$0 \leq s < m$$

Dette gir oss følgende valg for $i, j \, og \, k$

i	j	k
$s-2$	$s-2$	$s-1$
$s-2$	$s-1$	$s-2$
$s-1$	$s-2$	$s-2$

Med samme forutsetning at $\tau_2 = 2\tau_1 \, der \, \tau_1 = \tau$ kan vi sette opp følgende ligning

$$\begin{aligned} A_e &= \alpha^{2^{s-2}\tau+2^{s-2}*2\tau} \\ &+ \alpha^{2^{s-1}\tau+2^{s-2}*2\tau} \\ &+ \alpha^{2^{s-2}\tau+2^{s-1}*2\tau} \\ A_e &= \alpha^{2^{s-2}\tau+2^{s-1}\tau} \\ &+ \alpha^{2^{s-1}\tau+2^{s-1}\tau} \\ &+ \alpha^{2^{s-2}\tau+2^{s}\tau} \end{aligned}$$

Med en omskrivning av dette resultatet sitter vi igjen med

$$\begin{aligned} A_e &= \alpha^{\tau(2^{s-1}+2^{s-2})} + \alpha^{\tau 2^s} + \alpha^{\tau(2^{s-2}+2^s)} \\ &= \alpha^{3*2^{s-2}\tau} + \alpha^{4*2^{s-2}\tau} + \alpha^{5*2^{s-2}\tau} \\ &= 0^? \end{aligned}$$

For enkelthets skyld og for å skaffe oss en bedre oversikt gjør vi følgende erstatning, $\beta = \alpha^{2^{s-2}\tau}$. Vi sitter da igjen med uttrykket

$$\begin{aligned} A_e &= \beta^3 + \beta^4 + \beta^5 \\ &= \beta^3(\beta^2 + \beta + 1) \\ &= 0 \end{aligned}$$

Fra tidligere resultater, se forrige avsnitt, vet vi at $\beta^3 \neq 0$. Vi vet og at $\beta^2 + \beta + 1 \, \epsilon \, GF[2^2]$. I tillegg kan polynomet skrives på formen

$$\beta^2 + \beta + 1 = \frac{\beta^3 + 1}{\beta + 1}$$

Siden da $\beta^2 + \beta + 1$ deler $\beta^3 + 1$ vil $\beta^3 = 1$. Det vil si at vi har

$$\alpha^{3*2^{s-1}\tau} = 1$$

Av dette får vi at

$$2^{s-1} * 3\tau \equiv 0 \mod (2^m - 1)$$

Ettersom 2^{s-1} ikke vil kunne sørge for dette resultatet, får vi

$$3\tau \equiv 0 \mod (2^m - 1)$$

Dette gir to tilfeller for τ. De tilfeller der m er jamn, og de tilfeller der m er odde.

$$\begin{aligned} m\,jamn &\Rightarrow \tau \equiv \frac{2^m - 1}{3} \mod (2^m - 1) \\ m\,odde &\Rightarrow \tau \equiv 0 \mod (2^m - 1) \end{aligned}$$

Det første tilfellet er uaktuelt da det faktum at m ikke kan være jamn for at graden til våre generatorpolynomer skal ha primtalls-grad. Det andre tilfellet er uaktuelt etter argumentasjonen i forrige avsnitt og vi har vist at i dette spesialtilfelle kan vi få en høyere verdi for den nedre grense av den lineære kompleksiteten. Det vil si at vi kan bruke formelen (4.6) i dette tilfellet.

4.3 Andre tilfeller

Resultatfilene våre gir oss en indikasjon på at det må eksistere flere tilfeller av maksimal lineær kompleksitet enn det som hittil har vært vist. Det er fremdeles mange skift som gir dette som ikke er undersøkt. Det en kan se på her er hvilken relasjon det er mellom τ_1 og τ_2. Er det andre tilfeller av relasjoner mellom dem enn det som er beskrevet tidligere i kapitlet, så må det være mulig å identifisere dem.

Ved en inngående analyse av våre resultater mener vi å kunne, i tillegg til forrige avsnitt, finne data for at om man velger τ_2 på formen

$$\tau_2 = 2^n\tau_1,\ 2^n\tau_1 \leq (2^m - 1)$$

så vil dette gi samme resultat for den lineære kompleksiteten som for valgene beskrevet i forrige avsnitt. Skulle dette være tilfelle, vil det skaffe oss en mye større utvidelse av teorien beskrevet i forrige avsnitt og utvide valgmulighetene for τ_1 og τ_2. Dette vil kunne gjøre resultatene mye mer anvendelige.

Det skulle være mulig å bevise dette for vårt tilfelle. En kan gjenta prosessen for det en brukte i forrige tilfelle, men og ta med bevis for de med vekt lik tre da en ikke har noe teori om dette fra før. Det viser seg allikevel at dette er vanskelig. Der hvor man tidligere har opplevd kanselleringer i utregninger, blir det her heller mer komplekst. Alternativ nummer to er at en kan se på dette ved hjelp av induksjon, ettersom det vil fra steg til steg være en utvidelse med faktor på to. Heller ikke denne bevisformen er enkle å gjennomføre da en ikke kan klart finne en formel som det er mulig å utvide slik at bevisformen passer inn.

Selv om en her ikke har klart å finne en tilfredsstillende metode for å få bevist korrektheten av våre antagelser, så vil vi allikevel gi en hypotese for dette. Dette fordi det kan finnes andre som kan gjennomføre dette, eller som vil drive mer forskning på akkurat dette emnet.

Hypotese 4.1 *La $s_t, s_{t+\tau_1}, s_{t+\tau_2}$være tre distinkte faser av samme maksimalsekvens, generert av et skiftregister hvis generatorpolynom g(x)s grad, m, har primtalls-grad. Hvis en velger ens parametre, $\tau_1$og τ_2, på formen*

$$\tau_2 = 2^n \tau_1, 1 \leq n < m$$

vil en være garantert en en lineær kompleksitet på formen

$$\sum_{i=1}^{3} \binom{m}{i}$$

Denne formelen er tidligere vist for mindre spesialtilfeller og for produktet av to distinkte faser av samme sekvens.

4.4 Videre forskning

I dette avsnittet følger det to forslag til videre arbeid innen dette emnet. Det ene tar for seg beviset av hvorfor det eksisterer en strengere verdi, enn den nedre grensen beskrevet i (2.5). Forslag nummer to omhandler den naturlige utvidelsen av problemet til å dreie seg om fritt valg av antall skift en velger å bruke for å kunne oppnå høyere verdier både av kompleksitet og sekvenslengde

4.4.1 Bevis av nedre grense

Som beskrevet i kapittel (3.3), har vi funnet at polynomer av primtalls-grad kan garantere en høyere nedre skranke enn det som er beskrevet i tidligere litteratur om emnet. Denne litteraturen er beskrevet i avsnittene (2.5) og (2.4). Her blir det bevist to viktige forutsetninger for vår forskning:

1. For produkter av originalsekvens og et skift av denne er man garantert en nedre lineær kompleksitet på $\sum_{i=1}^{2}\binom{m}{i}$, der m er graden til polynomet.

2. For produkter av k ekvidistante faser av en sekvens, er man garantert en nedre grense på den lineære kompleksiteten på $L(z) \geq \binom{m}{k}$, der m er graden til polynomet.

Det vi mener å ha funnet vist statistisk i våre data er at den nedre grensen for lineær kompleksitet, etter å ha innført en begrensning på visse uaktuelle skift, kan strammes inn til $L(z) \geq \sum_{i=2}^{3}\binom{m}{i}$, der m er graden til generatorpolynomet, og vil med det fremsette følgende hypotese:

Hypotese 4.2 *La $s_t, s_{t+\tau_1}, s_{t+\tau_2}$være tre distinkte faser av samme maksimalsekvens, generert av et skiftregister hvis bindingspolynom har primtalls-grad, m. La produktsekvensen til disse tre ha betegnelsen z. Da vil produksekvensen ha nedre grense for den lineære kompleksiteten, $L(z)$, med unntak der produktsekvensen er nullsekvensen, på*

$$L(z) \geq \sum_{i=2}^{3}\binom{m}{i}$$

4.4.2 Se på flere skift

Den neste naturlige byggestenen innenfor dette emnet ville være å se på resultater for sekvenser hvor man bruker flere enn tre skift. Etter det vi kan se vil resultatene kunne være like gode for andre valg av skiftmengder. Hvorfor vil det være forskjell i utvidelsen fra tre til flere, enn det var fra to til tre. Det er ingenting i våre data som tyder på at de resultatene vi har kommet fram til ikke skal kunne gjelde ved en utvidelse. En ville da kunne se på produkter av mange skift og med dette få en stor lineær kompleksitet samtidig som sekvenslengden ville øke drastisk. Dette er som nevnt i (1) noe av det en vil oppnå ved design av strømchiffersystemer. Dette leder frem til en ny hypotese:

Hypotese 4.3 *La scenarioet være som beskrevet i (Hypotese 4.1), men med en utvidelse av mengden av skift som en velger. Da vil den lineære kompleksiteten til produktsekvensen definert av k skift være på samme form som for tre skift*

$$\sum_{i=2}^{k}\binom{m}{i}$$

MERKNAD:
Antall slike skift bør ikke overstige graden til generatorpolynomet.

Kapittel 5

Oppsummering og konklusjon

Vi har sett på produkter av maksimalsekvenser der en har multiplisert en sekvens med 2 vilkårlige, men ikke like skift av seg selv. I de numeriske reslutatene har vi funnet at det kan gis sterkere skranker på kompleksiteten til en slik sekvens, med visse forutsetninger, enn det som tidligere er vist. Vi har bevist en forbedring av en skranke gitt av Rueppel [2] med våre parametre. Det er også en utvidelse av Keys resultat [4] med visse begrensninger. Vi har også kunnet komme fram til hypoteser fra disse dataene, hypoteser som skal kunne bevises.

Disse resultatene viser at innenfor dette emnet er det mye som kan og bør forbedres. Ved å ta utgangspunkt i vårt arbeid kan en finne sterkere skranker for den lineære kompleksiteten til produkter av maksimalsekvenser, noe som kan gjøre dem mere attraktive innenfor kryptografiske systemer.

Bibliografi

[1] A.J. Menenez, P.C. van Oorschot, S.A. Vanstone.”Handbook of Applied Cryptography”, *CRC Press,*1997

[2] R.A. Rueppel.”Analysis and Design of Stream Ciphers”,*Springer-Verlag,*1986

[3] D.R. Stinson.”Cryptography, Theory and Practice”,*CRC Press,*1995

[4] E.L. Key.”An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators”, *IEEE Trans. Inform. Theory,* vol. IT-22, nov. 1976

[5] James L. Massey.”Cryptography: Fundamentals and Applications(Copies of Transparencies)”, *Advanced Technology Seminars,* 1993.

Tillegg A

Programkode

A.1 Hovedprogrammet

```
#include ¨ BerleBin.h¨
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#define POLYDEG 13 Variabel som forteller hvilken grad polynomet har
#define VERDI 377 Variabel som definerer forventet kompleksitet

void generate(char init[],char seq[],char poly[],int initl,int seql,int polyl);
void shift(char seq[],int seql,int antall);
void printarray(char seq[],int seql);                                    10
void copyarray(char source[],char dest[],int length);

 Metode som genererer en sekvens utfra et gitt shiftregister
    og en starttilstand

void generate(char init[],char seq[],char poly[],int initl,int seql,int polyl) {
  int bit = 0;
  for (int i = 0; i < seql; i++){
    løkke som går gjennom hvert element i den nye sekvensen
                                                                         20
    for (int j = 0; j < POLYDEG; j++){
      løkke som går gjennom polytab for  finne ut hvor det skal tappes

      if (poly[j]==1) {
        bit = bitˆinit[j];   er settes s(t+n) biten for hver tapping
      }
    }
    seq[i] = init[0];              Setter bit i sekvensen som det første i init sekvensen
    for (int k = 0;k < initl−1; k++) {
```

```
      init[k] = init[k+1];          Flytter init en bort, dvs skifter
    }
    init[initl-1] = bit;
    bit = 0;              Resetter s(t+n) biten
  }
}

Metode som kopierer en tabell av en gitt lengde
void copyarray(char source[],char dest[],int length) {
  for (int i = 0; i < length; i++) {
    dest[i] = source[i];
  }
}

Metode som shifter en sekvens antall-ganger
void shift(char seq[],int seql,int antall) {
  for (int j = 0; j < antall; j++) {
    int startbit = seq[0];          Tar vare på frste biten
    for (int i = 0; i < seql - 1; i++) {
      seq[i] = seq[i+1];            Bit i+1 settes til bit i
    }
    seq[seql - 1] = startbit;          Setter siste biten som opprinnelig var den første
  }
}

Metode for skrive ut innholdet i en tabell
void printarray(char seq[],int seql) {
  for ( int i = 0; i < seql; i++)
    printf("%d ",seq[i]);
  printf("\n");
}

int main(){
  int a = -1;
  int b = -1;
  char polytab[POLYDEG] = {1,1,0,1,1,0,0,0,0,0,0,0,0};      Anngir hvor det skal tappes
  int length = ((1 << POLYDEG) - 1)2;  length = (2^deg(pol)-1)  2
  char inittab[POLYDEG] = {1,0,0,0,0,0,0,0,0,1,0,0,0};        Setter initverdien i shiftregisteret

  char seqtab = (char)malloc(lengthsizeof(char));
  char seqtab2 = (char)malloc(lengthsizeof(char));
  char seqtab3 = (char)malloc(lengthsizeof(char));
  char multitab = (char)malloc(lengthsizeof(char));

  generate(inittab,seqtab,polytab,POLYDEG,length,POLYDEG);
  copyarray(seqtab,seqtab2,length);
  copyarray(seqtab,seqtab3,length);
```

```
    Her skiftes tabellene rundt.
       Tabell 3 skiftes aldri likt med tabell2
  for (int i = 1; i < length2; i++) {                                    80
    shift(seqtab2,length,i);
    for (int j = i+1; j < length2; j++) {
      shift(seqtab3,length,j);
      for (int l = 0; l < length; l++){
        multitab[l] = seqtab[l]&seqtab2[l]&seqtab3[l];
      }
      int a = -1;
      int b = -1;

      FinnKompleksitet(multitab,&a,&b,length,VERDI,i,j);                 90
      copyarray(seqtab,seqtab3,length);
      fflush(stdout);
    }
    copyarray(seqtab,seqtab2,length);
  }
  return 0;
}

                                                                         100
```

A.2 Berlekamp-Massey algoritme

```
#define PolynomMax 5000
#include <stdio.h>

void PolyMulti(char ,char ,char ,char ,char );
void Polyaddi(char ,char ,char ,char ,char );
void PolySettLik(char ,char ,char ,char );
int  FinnDelta(char ,char ,char ,char );
void FinnKompleksitet(char ,int ,int ,int,int,int,int);



                                                                    10

void PolyMulti (char poly1[],char poly2[],char poly3[],int L,int x)
{
 int i,j,max;

 max=(L + x + 2);
 printf(Ï=%d og x=%d og max=%d \n, L, x,max);
 for (i=0;i<=max;i++)
 {
  poly3[i]=0;                                                       20
  for (j=0;j<=i;j++)
  {
   poly3[i]=(poly3[i]+poly1[i−j]poly2[j]) % 2;
  }
 }
}



void PolyAddi (char poly1[],char poly2[],char poly3[],int L,int x)  30
{
 int i,max;

 max=(L + x + 2);
 for (i=0;i<=max;i++)
 {
  poly3[i]=(poly1[i]+poly2[i]) % 2;
 }
}
                                                                    40


void PolySettLik (char poly1[],char poly2[],int L,int x)
{
```

```
 int i,max;

 max=(L + x +2);
 for (i=0;i<=max;i++)
 {
  poly2[i]=poly1[i];                                                    50
 }
}



int FinnDelta (char Poly[],char Sekvens[],int L,int n)
{
 int i,sum;

 sum=Sekvens[n];                                                        60
 for (i=1;i<=L;i++)
 {
  sum=sum+Poly[i]   Sekvens[(n−i)];
 }
 printf(Ï finndelta er sum=%d\n,sum % 2);
 sum=(sum % 2);
 return(sum);
}

                                                                        70

void FinnKompleksitet(char Sekvens[],int MaxL,int AntMaxL,int SLength,int verdi, int shift1,int shift2)
{
 int n,L,x,delta,j,k,z;
 char PolyC[PolynomMax],PolyCs[PolynomMax],PolyT[PolynomMax],Temp[PolynomMax];
 char Temp2[PolynomMax];
 int a= shift1;
 int b= shift2;
 int c= verdi;
 L=0;x=1;                                                               80
 for (j=1;j<PolynomMax;j++)
 {
    PolyCs[j]=0;
    PolyC[j]=0;
 }
 PolyC[0]=1;PolyCs[0]=1;
 for (n=0;n<SLength;n++)
 {
    delta=FinnDelta(PolyC,Sekvens,&L,&n);
    if (delta == 0)                                                     90
        x=x+1;
```

```
    else
    {
      PolySettLik(PolyC,PolyT,&L,&x);
      for (j=0;j<x;j++)
           Temp[j]=0;
      Temp[x]=1;
      for (j=x+1;j<PolynomMax;j++)
           Temp[j]=0;
      PolyMulti(Temp,PolyCs,Temp2,&L,&x);                          100
      PolyAddi(PolyC,Temp2,Temp,&L,&x);
      for (j=0;j<PolynomMax;j++)
           PolyC[j]=Temp[j];
      if (2L > n)
          x=x+1;
      else
      {
         printf(Öppdaterer L,x,PolyCs \n);
        L=n+1-L;
        x=1;                                                       110
        PolySettLik(PolyT,PolyCs,&L,&x);
         printf(Ï else staement oppdateres x til %d \n,x);
      }
    }
  }
  if (L > MaxL)
  {
      MaxL=L;
      AntMaxL=1;
      if (MaxL!=c){                                                120
        printf(Ï= %d\n¨,a);
        printf(Ï= %d\n¨,b);
        printf(Ïen nye max kompleksiteten er %d .\n¨,L);
      }
  }
  else if (L == MaxL)
  {
    AntMaxL=AntMaxL + 1;
    printf(Ïangering av Max Komplex.\n¨);
  }                                                                130
}
```

Tillegg B

Tabeller

B.1 Starttilstander

Her følger en liste over hvilke polynomer og starttilstander som er brukt under genereringen av sekvensene.

Grad	Bindingspolynom	Starttilstand
3	$x^3 + x + 1$	100
4	$x^4 + x + 1$	0001
5	$x^5 + x^2 + 1$	10010
6	$x^6 + x + 1$	100001
7	$x^7 + x + 1$	1000000
8	$x^8 + x^4 + x^3 + x^2 + 1$	00010100
9	$x^9 + x^4 + 1$	100001000
10	$x^{10} + x^3 + 1$	0000000100
11	$x^{11} + x^2 + 1$	10000000010
12	$x^{12} + x^6 + x^4 + x + 1$	000000000001
13	$x^{13} + x^4 + x^3 + x + 1$	1000000001000

B.2 Iterasjonsverdier

Tabellen under beskriver utviklingen i i kompleksitet i Berlekamp-Masseys algoritme. Som tallene forteller ser en at det ikke er gode muligheter under normale forhold å generere komplette filer for alle grader av polynomer. Antall iterasjoner gjelder for et skift av sekvensen i verste tilfelle.

Polynomgrad	Antall iterasjoner
3	196
4	900
5	3844
6	15876
7	64516
8	260100
9	1044484
10	4186116
11	16760836
12	67076100
13	268369924

B.3 Faktorisering av $2^m - 1$

Her følger en oversikt over om $2^m - 1$ kan faktoriseres eller ikke. Dette er viktig for valg av polynom med grad m, der m er primtall

$2^m - 1$	Faktorer
$2^3 - 1$	7
$2^5 - 1$	31
$2^7 - 1$	127
$2^{11} - 1$	23x89
$2^{13} - 1$	8191
$2^{17} - 1$	131071
$2^{19} - 1$	524287
$2^{23} - 1$	47x178481
$2^{29} - 1$	233x1103x2089
$2^{31} - 1$	2147483647

B.4 Resultattabeller

De påfølgende resultattabellene er formet ved at de er delt opp i 2 deler. En for null-resultat, og en for ikkenull-resultat. De skiftene som gir den maksimale verdi, det vil si de som beholder alle sine røtter er ikke listet opp. Vi har altså bare listet opp avvikene.

B.4.1 Grad lik 3

Skift 1	Skift 2	L(s)
1	3	0
2	6	0
4	5	0

B.4.2 Grad lik 5

Skift 1	Skift 2	L(s)
1	18	0
2	5	0
3	29	0
4	10	0
6	27	0
7	22	0
8	20	0
9	16	0
11	19	0
12	23	0
13	14	0
15	24	0
17	30	0
21	25	0
26	28	0

B.4.3 Grad lik 7

Resultat lik 0

Skift 1	Skift 2	L(s)
1	7	0
2	14	0
3	63	0
4	28	0
5	54	0
6	126	0
8	56	0
9	90	0
10	108	0
11	87	0
12	125	0
13	55	0
15	31	0
16	112	0
17	43	0
18	53	0
19	29	0
20	89	0
21	57	0
22	47	0
23	82	0
24	123	0
25	105	0
26	110	0
27	66	0
30	62	0
32	97	0
33	77	0
34	86	0

Skift 1	Skift 2	L(s)
35	109	0
36	106	0
37	46	0
38	58	0
39	100	0
40	51	0
41	75	0
42	114	0
44	94	0
45	68	0
48	119	0
49	122	0
50	83	0
52	93	0
59	104	0
60	124	0
61	88	0
64	67	0
65	95	0
69	107	0
70	91	0
71	79	0
72	85	0
73	78	0
74	92	0
76	116	0
80	102	0
81	118	0
84	101	0
96	111	0
98	117	0
99	103	0
113	115	0
120	121	0

Resultat lik 56

Skift 1	Skift 2	L(s)
1	3	56
1	15	56
1	18	56
1	31	56
1	63	56
1	94	56
2	6	56
2	30	56
2	36	56
2	61	56
2	62	56
2	126	56
3	7	56
3	15	56
3	31	56
4	12	56
4	60	56
4	72	56
4	122	56
4	124	56
4	125	56
5	9	56
5	25	56
5	44	56
5	77	56
5	94	56
5	105	56
6	14	56
6	30	56
6	62	56
7	15	56
7	31	56
7	63	56
7	75	56
7	93	56

Skift 1	Skift 2	L(s)
8	17	56
8	24	56
8	117	56
8	120	56
8	121	56
8	123	56
9	26	56
9	37	56
9	45	56
9	47	56
9	55	56
9	64	56
9	77	56
9	93	56
9	109	56
9	113	56
9	119	56
10	18	56
10	27	56
10	50	56
10	61	56
10	83	56
10	88	56
11	33	56
11	38	56
11	58	56
11	77	56
12	28	56
12	60	56
12	124	56
13	68	56
13	110	56
14	23	56
14	30	56
14	59	56

Skift 1	Skift 2	L(s)
14	62	56
14	126	56
15	63	56
15	72	56
15	106	56
16	34	56
16	48	56
16	107	56
16	113	56
16	115	56
16	119	56
17	30	56
17	35	56
17	80	56
17	84	56
17	85	56
17	89	56
17	91	56
17	93	56
17	117	56
17	118	56
17	126	56
18	27	56
18	52	56
18	59	56
18	74	56
18	90	56
18	91	56
18	94	56
18	99	56
18	110	56
18	111	56
19	59	56
19	69	56
19	80	56

Skift 1	Skift 2	L(s)
19	91	56
19	102	56
19	107	56
20	36	56
20	39	56
20	49	56
20	54	56
20	100	56
20	122	56
21	36	56
21	93	56
22	27	56
22	66	56
22	76	56
22	116	56
23	59	56
24	56	56
24	120	56
24	121	56
25	44	56
25	54	56
25	91	56
25	93	56
25	94	56
26	93	56
27	47	56
27	76	56
27	116	56
28	46	56
28	60	56
28	118	56
28	124	56
28	125	56
29	69	56
29	80	56

Skift 1	Skift 1	L(s)
29	102	56
29	107	56
30	85	56
30	126	56
31	63	56
32	68	56
32	87	56
32	96	56
32	99	56
32	103	56
32	111	56
33	34	56
33	38	56
33	51	56
33	58	56
33	87	56
34	41	56
34	43	56
34	51	56
34	55	56
34	59	56
34	60	56
34	70	56
34	107	56
34	109	56
34	125	56
35	91	56
36	53	56
36	54	56
36	55	56
36	61	56
36	71	56
36	93	56
36	95	56
36	104	56

Skift 1	Skift 2	L(s)
36	118	56
37	55	56
38	55	56
38	77	56
38	87	56
38	118	56
39	49	56
39	89	56
39	122	56
40	72	56
40	73	56
40	78	56
40	98	56
40	108	56
40	117	56
41	109	56
42	59	56
42	72	56
43	60	56
44	54	56
44	105	56
45	113	56
46	118	56
47	64	56
47	66	56
47	76	56
47	116	56
48	112	56
48	113	56
48	115	56
49	89	56
49	100	56
50	55	56
50	59	56
50	61	56
50	88	56
50	108	56

Skift 1	Skift 1	L(s)
51	73	56
51	78	56
51	98	56
51	117	56
52	59	56
53	71	56
54	94	56
54	105	56
55	59	56
55	70	56
55	95	56
55	97	56
55	114	56
55	118	56
56	92	56
56	109	56
56	120	56
56	121	56
56	123	56
57	91	56
57	112	56
58	77	56
58	87	56
59	72	56
59	91	56
59	125	56
60	125	56
61	83	56
61	108	56
62	126	56
63	72	56
63	110	56
64	65	56
64	71	56
64	79	56

Skift 1	Skift 1	L(s)
64	95	56
65	67	56
65	71	56
65	79	56
66	68	56
66	76	56
66	102	56
66	116	56
67	71	56
67	79	56
67	95	56
67	101	56
67	110	56
68	82	56
68	86	56
68	87	56
68	91	56
68	102	56
68	110	56
68	118	56
68	120	56
68	123	56
69	80	56
69	102	56
71	95	56
72	81	56
72	106	56
72	108	56
72	109	56
72	110	56
72	122	56
73	93	56
73	98	56
73	109	56
73	117	56

Skift 1	Skift 1	L(s)
74	110	56
75	93	56
76	109	56
76	110	56
77	87	56
78	98	56
78	117	56
79	95	56
80	89	56
80	107	56
81	109	56
82	91	56
83	88	56
83	108	56
84	118	56
86	120	56
88	108	56
89	100	56
89	122	56
90	99	56
91	93	56
91	111	56
91	112	56
92	109	56
93	109	56
93	126	56
94	105	56
96	97	56
96	99	56
96	103	56
97	99	56
97	103	56
97	111	56
97	114	56
99	111	56

Skift 1	Skift 1	L(s)
100	110	56
100	118	56
100	122	56
101	110	56
102	107	56
103	111	56
104	118	56
109	110	56
109	119	56
110	118	56
112	113	56
112	115	56
112	119	56
113	119	56
115	119	56
118	123	56
120	123	56
121	123	56
124	125	56