Kryptering eller kaos!

Af Christel Bach

Om denne artikel:
I artiklen "Security of Public Key Cryptosystems based on Chebysjev Polynomials" af Bergamo,D'Arco, De Santis og Kocarev (BASK) gennemgåes anvendelse af kaotisk system som grundlag for kryptering, i artiklen anvendes chebyshevs polynomier som det kaotiske system.
I denne artikel vil jeg gennemgå flere af de centrale problemstillinger ved denne anvendelse, ligesom jeg vil se på hvad der generelt karaktiserer kaotiske systemer som grundlag for kryptering.
Artiklen er skrevet som eksamensopgave juni 2005 i kryptologi A ved Århus Universitet

Indledning

De fleste kryptosystemer folder sig ud i modulær aritmetik og baseres på såkaldt svære problemer. Det være sig alt fra logaritme-problemet til faktorisering af store tal. Svære problemer vil sige at det i praksis har vist sig svært at vende en algoritme om. Det er let at gange to meget store primtal sammen, mens det er meget svært at faktorisere[1] et stort tal i primtal. Det er også let at opløfte et tal i en høj potens, mens det er svært[2] at regne ud hvilken potens der blev opløftet til på resultatet.
Man taler om at man anvender en såkaldt "trapdoor". En process der er irreversibel i praksis.

Når man interesserer sig for kaotiske systemer i krypteringsmæssig sammenhæng er det især fordi systemerne er meget følsomme for små forskelle i input ved mange iterationer.
Stor følsomhed[3] er en meget værdifuld egenskab ved kryptografiske algoritmer, da en ændring i input på bare en bit fx vil give et helt anderledes resultat. Denne følsomhed sikrer at hvis en opponent forsøger at afprøve systemet med forskellige startbetingelser for at se efter mønstre, vil det ikke virke, selv gæt der ligger meget tæt på det rigtige input vil ikke afsløre sig.
Det betyder altså at man ved anvendelse af kaotiske systemer som udgangspunkt kan skabe et kryptosystem med stor følsomhed.

I dette tilfælde har BASK[4] valgt at anvende Chebyshevs polynomier da disse ved rekursion udviser kaotiske egenskaber. Forfatterne konstruerer et system baseret på decimaltal og gennemgår hvordan dette system kan anvendes til kryptering og dekryptering, autentifikation og nøgleudveksling.
Artiklen gennemgår desuden hvordan systemet kan brydes og diskuterer kort hvordan decimaltals implementation sætter grænser for systemet og dets sikkerhed.

[1] Det er let at gange 3457 med 4049 og derfor se hvordan 13997393 er sammensat, men hvilke primtal kan f.eks 30168493 opløses i ?
[2] Hvis $a^p=N$, og a er forskellige fra 0 og 1, så er $p=\log_a(N)$
[3] K.M. Roskin and J.B. Casper, 'From Chaos to Cryptography'
[4] Bergamo,D'Arco, De Santis og Kocarev (BASK)

Kaotiske systemer generelt[5]

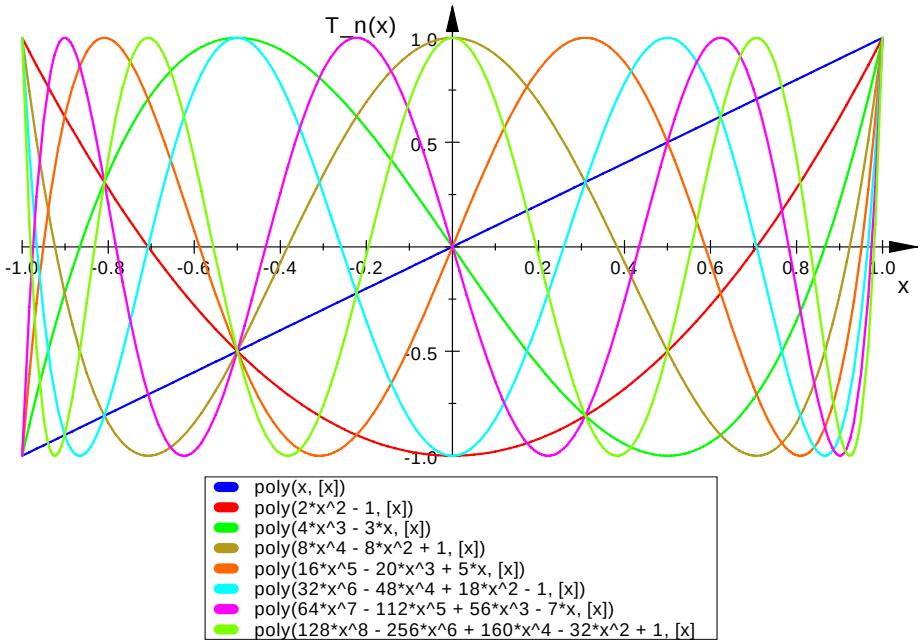
Et kaotisk system er et dynamisk system der udviser visse egenskaber.
Et eksempel på et sådant dynamisk system kan være populationsudvikling.
Hvis systemet kunne vokse uhæmmet for hver ny generation ville vi se eksponentiel vækst.
Formeringraten k bestemmer hvor voldsomt det sker, men indfører vi en begrænsning får vi tendenser i systemet. Systemet virker således at hver ny generation beregnes på baggrund af den forgående ved rekursion.
Ex.: Kvadratisk mapping: $P_{(n+1)=}kP_n(1-P_n)$
P er populationens størrelse i procent i en generation n. k er formeringsfaktoren og 1 repræsenterer en befolkning på 100%, altså optimalt antal individer i forhold til fødemængde – altså en begrænsning i vækst , systemet vil da stabiliseres hvis P =1, og populationen vokse hvis P <1 og falde hvis P>1.
Dette simple system udviser stor og uventet kompleksitet[6].
Et kaotisk system kan groft karakteriseres således: har sensitiv afhængighed af startbetingelser, kan ikke ”dekomponeres” i to undersystemer og har et element af regularitet, nemlig periodiske punkter med stor tæthed.

Introduktion til Chebyshevs polynomier

Chebyshevs polynomier er allestedsnærværende[7] i den matematiske analyse og dukker op de mest overraskende steder, og deres anvendelse er vidtspredte. Der er et antal forskellige slags Chebyshev polynomier. Fra første til fjerde ’art’ og såkaldt ’shiftede’ varianter af samme. De har alle tætte relationer til de trigonometriske funktioner sinus og cosinus. I artiklen refereres der som så mange andre steder kun til Chebyshevs polynomier, og det er da underforstået at der er tale om Chebyshevs polynomier af første art.
Disse polynomier udviser den ønskede kaotiske adfærd og er da anvendt i artiklen som grundlag for for et kryptosystem.
Chebyshevs polynomier af første art kan rekursivt defineres som :

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \text{ for n>2} \qquad T_0(x) = 1, T_1(x) = x \qquad (1.1)$$

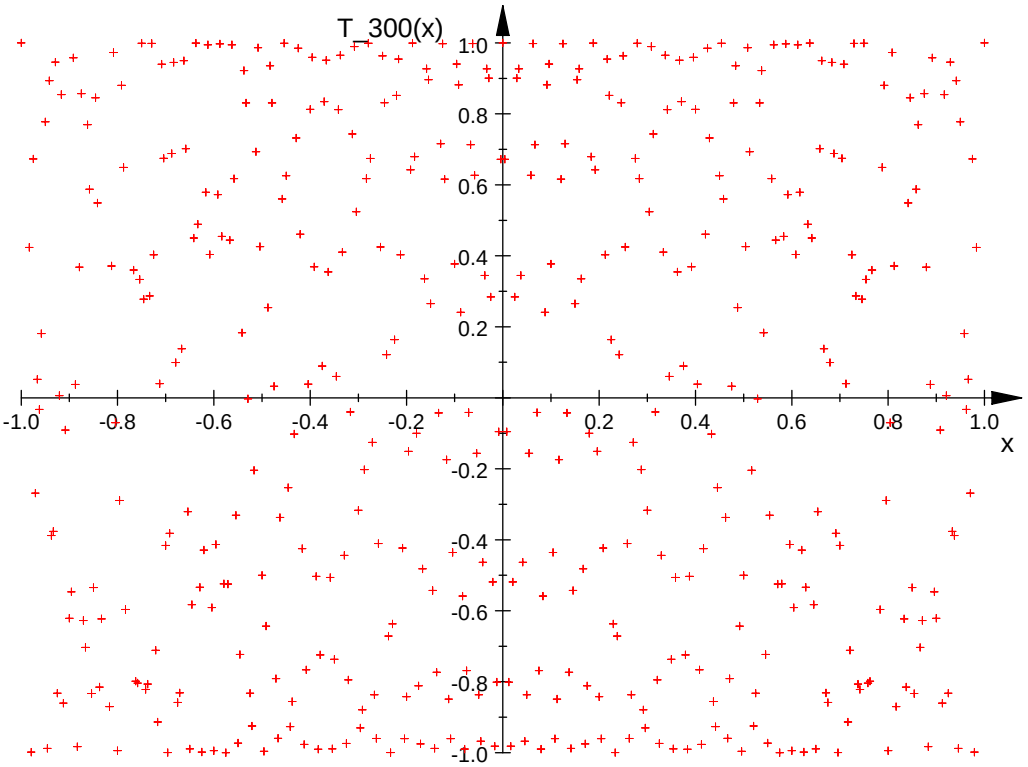
Herunder er illustreret de 8 første trin af rekursionen, dels hvordan polynomiet udvikler sig, dels en grafisk illustration af sammenhørende værdier af x og T_n (x) for disse trin.



[Fig 1]

[5] ‘An Introduction to Chaotic Dynamical Systems’, 2.ed. Robert L. Devaney
[6] http://mathworld.wolfram.com/LogisticMap.html
[7] ‘Chebyshev Polynimials’ J.C. Mason og D.C. Handscomb, Chapman&Hall/CRC 2003

Det ser jo ikke specielt kaotisk ud for små n, men hvis n øges meget forandrer billedet sig. Det kan ses på grafen herunder hvordan Chebyshevs polynomier udviser meget stor følsomhed for initialværdi ved mange iterationer, herunder er anvendt 300 iterationer og en præcision på 100 cifre. Følsomheden ses tydeligst når jeg som her har valgt at illustrere sammenhørende værdier af $x,T_{300}(x)$ med punkter med en inddeling på 0,01. Det samme billede tegner sig hvis man anvender mange flere iterationer sammenholdt med meget finere inddeling, svarende til finere "opløsning"



[Fig 2]

Scope

Chebyshevs polynomier er defineret som ovenstående i området [-1,1]. Hvis vi skal definere disse polynomier til at gælde uden for området benyttes en lineær transformation [8]således at $T_n(s)$ i området [a,b] gives ved

$$s = \frac{2x - (a+b)}{b-a}$$

Implementation og tidskomplexitet

Når vi man implementer Chebyshevs polynomier vil man hurtigt opdage at mange iterationer er meget tungt. En direkte implementation af algoritmen udføres i lineær tid. O(n) Dette kan dog forkortes til logaritmisk tidskompleksitet O(log n) ved at reducere rekursionen på følgende måde:

Det fremgår af (1.6) og Fig 1 at $T_{2n}(x) = T_2(T_n(x)) = 2T_n(x)^2 - 1$

Dette kan omskrives til $T_{2n}(x) = 2T_n(x)^2 - T_0$ og ved iteration har vi da at

$T_{2n+1}(x) = 2T_{n+1}(x)T_n(x) - T_1 = 2T_{n+1}(x)T_n(x) - x$

[8] 'Chebyshev Polynimials' J.C. Mason og D.C. Handscomb, Chapman&Hall/CRC 2003, (1.31)

Alternativ definition

Chebyshevs polynomier af første art kan også defineres ved hjælp af cosinus funktionen således:

$$T_n(x) = \cos n\theta \text{ hvor } x = \cos\theta . \quad (1.2)$$

Det betyder at for værdier af x i intervallet [-1,1] får vi korresponderende værdier af θ mellem $[0, \pi]$. Det betyder også at vi i samme interval kan anvende definitionen:

$$T_n(x) = \cos(n * \cos^{-1}(x)) \quad (1.3)$$

idet vi har at når $x = \cos\theta$ er $\cos^{-1}(x) = \theta$ for x vædier mellem [-1,1]

Vi kan indse at denne definition stemmer overens med (1.1) ved at betragte den trigonometriske additionsformel for cosinus:

$$\cos(a \pm b) = \cos a \cos b \pm \sin a \sin b$$

Vi har da at $\cos(n\theta \pm \theta) = \cos(n\theta)\cos(\theta) \pm \sin(n\theta)\sin(\theta)$, ved at addere disse ligninger får vi:

$\cos(n\theta + \theta) + \cos(n\theta - \theta) = 2\cos(n\theta)\cos(\theta) + \sin(n\theta)\sin(\theta) - \sin(n\theta)\sin(\theta)$

Vi får da ved reduktion og distribution

$\cos((n+1)\theta) + \cos((n-1)\theta) = 2\cos(n\theta)\cos(\theta)$ hvilket er det samme som

$$\cos((n+1)\theta) = 2\cos(n\theta)\cos(\theta) - + \cos((n-1)\theta) \quad (1.4)$$

og erstattes her $x = \cos\theta$ og $T_n(x) = \cos n\theta$

ser vi at det er det samme som

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (1.5)$$

Der desuden gælder følgende for krypteringssystemet afgørende egenskab:

$$T_n(T_m(x)) = T_{nm}(x) \quad (1.6)$$

Det følger direkte af polynomiets relation til cosinus (3):

$T_n(T_m(x)) = \cos(n\cos^{-1}(\cos(m\cos^{-1}(x)))) = \cos(nm\cos^{-1}(x)) = T_{nm}(x)$

Kryptering generelt:

Det aktuelle kryptosystem er et public key system. Et public key system giver sikker kommunikation over en åben linie uden deling af hemmelig nøgle via offentlige og private nøgler. Alle udstyres med såvel offentlige som private nøgler og sikkerheden i systemet er baseret på den antagelse at man ikke kan beregne den private nøgle ud fra offentlig nøgle og krypteret besked.

Der er udviklet mange public key systemer der alle er baseret på trapdoors og flere kan overordnet beskrives således

$$X_{n+1} = (X_n)^p \pmod N \quad (1.7)$$

ElGamal implementerer dette således at N er et primtal, X_0 er generator i gruppen Z_N^* , p vælges så den ligger mellem 1 og N-2. Trapdooren er altså logaritmeproblemet.

Et eksempel på kryptering med ElGamal:
Vi har kendt primtal p=47 og generator g=2. Alice vælger som sin hemmelige nøgle a tallet 34 og beregner hendes offentlige nøgle til A= 2^{34}= 17179869184.
Bob vil sende beskeden 42 til Alice og vælger et k<p, k=13. Han beregner da
$c_1 = g^k \bmod p = 2^{13} \bmod 47 = 14$ og $c_2 = A^k * m \bmod p = 17179869184^{13} * 42 \bmod 47 = 28$. Han

sender altså (14,28) tl Alice. Hun rekonstruerer beskeden ved at beregne
$m = c_1^{-a} * c_2 \bmod p = 14^{-34} * 28 \bmod 47 = 42$. Dette er korrekt fordi:
$c_1^{-a} * c_2 \bmod p = (g^k)^{-a} * A^k * m = g^{-a*k} * A^k * m = (g^a)^{-k} * A^k * m = A^{-k} * A^k * m = m$

RSA implementerer (1.7) således at N=PQ, hvor P og Q er to store og forskellige primtal. P vælges mellem 1 og φ hvor φ =(P-1)(Q-1) når φ og p ikke har nogen fælles divisorer, dvs gcd(φ,p)=1. X_0 er her beskeden der krypteres. Trapdooren er her primtalsfaktorisering.

Begge systemer anvender den meget afgørende egenskab $(X^p)^q = X^{pq} (\bmod N)$
Kryptering med Chebyshev kan ses som en variant af disse kryptosystemer. I modsætning til ElGamal er det ikke blot en eksponent der skal findes men beregning af et polynomiums orden og dette synes umiddelbart som værende et endnu sværere problem.

Generelt blev polynomial rekonstruktion (PR)[9] introduceret som et nyt svært problem i 1999 af Kiayias og Yung. Problemstillingen er siden studeret mangesidigt og adskillige kryptografiske systemer baseret på PR er siden foreslået og undersøgt.
F.eks har Augot og Finasz[8] skitseret en interessant model hvor Reed-Solomon koder fungerer som basis for sådan et system.

Men hvor f.eks ElGamal finder sted i et modulært system og logaritmeproblemet derfor har en entydig løsning vil det efterfølgende fremgå at adskillige Chebyshev polynomier krydser de samme punkter , jf. Fig 1, og løsningerne dermed heller ikke entydige. Så selv om PR synes at være en fornuftig trapdoor er der en genvej til at bryde Chebyshev-systemet.

Krypteringssystemet med Chebyshev

Systemet består af følgende algoritmer:

Nøglegenerering

1. Alice vælger et stort heltal s
2. Hun vælger et tilfældigt tal x mellem -1 og 1 og beregner $T_s(x)$
3. Alice publicerer da hendes offentlige nøgle K_p som $(x,T_s(x))$ og hemmeligholder hendes private nøgle $K_s = s$

Kryptering

Bob ønsker at sende en besked til Alice og finder derfor hendes offentlige nøgle

1. Beskeden udtrykkes som et tal M mellem -1 og 1
2. Bob generer et stort heltal r
3. Herefter beregner han $T_r(x), T_{rs}(x) = T_r(T_s(x))$ og $X = M * T_{rs}(x)$
4. Den krypterede besked $C = (T_r(x), X)$ sendes da til Alice

Dekryptering

Alice finder klarteksten ved at udføre følgende beregning

1. Anvender hendes private nøgle s til at beregne $T_{rs}(x) = T_s(T_r(x))$
2. Finder M ved at beregne $M = X / T_{rs}(x)$

[9] "A Public Key Encryption Scheme Based on the Polynimial Reconstruction Problem" af Daniel Augot og Matthieu Finiasz, INRIA, Domaine de Voluceau

Som det ses er det den afgørende kommutationsegenskab (1.6) der er grundlaget for at dekryptering kan finde sted da $T_r(T_s(x)) = (T_{rs}(x)) = T_s(T_r(x))$

Eksempel 1:
Bob ønsker at sende beskeden M=0.1235711131719 til Alice
Alice vælger derfor sin private nøgle s=42 og vælger sig et tilfældigt tal
x=-0.11235813213455 og beregner T_{42}(-0.11235813213455)= -0.016637774716629
Bob vælger tilfældigt et tal r=67 og beregner
X=M*(T_{67}(T_{42}(x)))= 0.1235711131719 * T_{67}(-0.016637774716629)= 0.11094399380442 og
T_{67}(x)=T_{67}(-0.11235813213455)= 0.9523162482865. Han sender derfor beskeden
(0.9523162482865, 0.11094399380442) til Alice
Hun dekrypterer beskeden ved at beregne:
M=0.11094399380442/T_{42}(0.9523162482865) og får korrekt at M=0.1235711131719

Bruddet

Svagheden ved Chebyshev systemer er at sammenhørende værdier mellem et polynomium og en x værdi ikke er unik. Der findes altså flere forskellige polynomier der krydser det samme punkt, således at $T_a(x) = T_b(x)$, Dette ses tydeligt på den grafiske repræsentation hvor polynomiernes ses at have skæringspunkter med hinanden.
Vi kan altså derfor finde et r' således at $T_{r'}(x) = T_r(x)$
En angriber der derfor har opsnappet beskeden $C = (T_r(x), X)$ fra Bob til Alice og har fundet Alices offentlige nøgle (x,T_s(x)) vil da kunne afsløre beskeden ved at beregne $T_{r'}(T_s(x))$, idet vi jo har at

$$M = X / T_{rs}(x) = X / T_{sr}(x) = X / T_s(T_r(x)) = X / T_s(T_{r'}(x)) = X / T_{sr'}(x)$$
$$= X / T_{r's}(x) = X / T_{r'}(T_s(x))$$

Der resterer altså blot at finde et sådan r'.
I artiklen af BASK anfører de hvorledes man i en implementationssituation kan finde sådan et r'.
Et af problemerne ved at finde det teoretisk eksisterende r' er præcision. Hvis vi ser på et meget groft afrundet system vil vi kunne finde mange r' der opfylder
Kravet $T_{r'}(x) = T_r(x)$. Men jo finere opløsning vi insisterer på at regne med, jo vanskeligere er det at ramme præcist i praksis.

r' kan i følge BASK beregnes som :

$$r' = \pm \frac{\cos^{-1}(T_r(x)) + 2k'\pi}{\cos^{-1}(x)} \qquad (1.8)$$

Argumentet er som følger. Hvis r' har form som (1.8) i den positive variant kan vi da skrive

$$T_{r'}(x) = \cos(r'\cos^{-1}(x)) = \cos(\frac{\cos^{-1}(T_r(x)) + 2k'\pi}{\cos^{-1}(x)} * \cos^{-1}(x))$$
$$= \cos(\cos^{-1}(T_r(x)) + 2k'\pi)$$

Og vi får da yderligere reduktion ved hjælp af additionsformlen for cosinus:

$= \cos(\cos^{-1}(T_r(x))\cos(2k'\pi) - \sin(\cos^{-1}(T_r(x))\sin(2k'\pi)$

$= \cos(\cos^{-1}(T_r(x))$

$= T_r(x)$

Argumentet er tilvarende i den negative verison.
Hvis vi omvendt antager at $T_{r'}(x) = T_r(x)$ for visse $r' \in \mathrm{N}$

Så har vi at $T_{r'}(x) = \cos(r' * \cos^{-1}(x)) = T_r(x)$

Anvender vi da $\cos^{-1}$ på begge sider af lighedstegnet får vi at

$\cos^{-1}(\cos(r' * \cos^{-1}(x))) = \cos^{-1}(T_r(x))$ (1.9)

Men hvis vi har et α så $\alpha = \cos^{-1}(\omega)$ og ønsker at finde alle β så $\cos(\beta) = \omega$ betyder det, da vi også har at $\cos(-\beta) = \cos(\beta)$, at alle løsninger skrives på formen $\beta = \alpha \pm 2\pi k$

Det betyder at (1.9) kun er sandt hvis og kun hvis

$r' \cos^{-1}(x) = \pm \cos^{-1}(T_r(x)) + 2\pi k'$

Vi får da at

$$r' = \pm \frac{\cos^{-1}(T_r(x)) + 2k'\pi}{\cos^{-1}(x)}$$

Hvis vi derfor forsøger at finde r' i eksempel 1 ovenfor har vi da at

$$r' = \frac{\cos^{-1}(\text{-}0.016637774716629) + 2k'\pi}{\cos^{-1}(\text{-}0{,}11235813213455)} = \frac{1.5874348692063 + 2\pi \mathrm{k}}{1.6833922201948}$$

Problemet er nu at r' skal være et naturligt tal. For ovenstående brøk med f.eks k=1 giver et r'=4.6754523883181.
I dette tilfælde vil vi altså skulle finde et helt tal k' således at r' ikke med det aktuelle antal cifre afviger fra et helt tal idet dette her kan kaldes vores "opløsning".

Vi er altså dermed nødt til at finde en heltalsløsning til de to ligninger for r'.
Hvis vi opdeler formlen for r' i to dele:

$a = \frac{\cos^{-1}(T_r(x))}{\cos^{-1}(x)}$ og $b = \frac{2\pi}{\cos^{-1}(x)}$ kan vi se hvorledes r' kan beregnes hvis vi kan finde en heltalsløsning $u \in \mathrm{N}$ til en af de to ligninger hvor $k \in \mathrm{Z}$:
$a + kb = u$ eller $-a + kb = u$ givet de to delbrøker a og b.

Disse ligninger løses i en implementationssituation moduleret i forhold til præcision i beregning.

Et eksempel på et forsøg på et brud:

Vi ønsker fortsat at bryde koden i eksempel 1. Vi har at r' er en heltalsløsning til (1.8)

Vi får at a= 0.9429976271 og b=3.732454761
Vi har da at løsningen skal findes i mængden

$\{\pm 0.9429976271 + 3.732454761 * \mathrm{k} \mid \mathrm{k} \in \mathrm{Z}\}$

Vi vælger kun at se på fraktionsdelen og får da

0.9429976271+0.732454761k=z, vi modulerer med B^L , hvor B er vores base og L antallet af cifre i vores beregning. Altså her 10^10 og får da
9429976271+ 7324547610k=$10^{10}z_1$
-9429976271+7324547610k=$10^{10}z_2$
Hvis vi forsøger at løse denne ligning modulært , altså modulus 10^{10}
$7324547610k \equiv 9429976271 (\mathrm{mod}\ 100000000000)$ finder vi ved nærmere undersøgelser ingen løsninger idet $\gcd(7324547610,\ 100000000000)=10$ og 10 deler ikke 9429976271. Det samme gør sig gældende for den anden ligning.
Havde vi derimod afrundet og regnet mod 10 havde vi sagtens fundet en løsning på f.eks ligningen
$7k \equiv 9 (\mathrm{mod}\ 10)$ der har [7] som løsning eller med en smule mere præcision
$73k \equiv 94 (\mathrm{mod}\ 100)$ der har [78] som løsning.

Et eksempel der 'virker'..

BASK har i artiklen beskrevet et eksempel på et brud hvor man tilsyneladende vælger at arbejde med 10 cifre.
Men ved nærmere eftersyn opererer de med exact præcision på Pi og får dermed en række akkumulerede fejl – desuden vælger de også i forbindelse med løsningen af kongruensen at sætte L=1 og dermed modulerer de kun med 10 – ergo kastes en masse betydende cifre væk i beregningerne, den får altså med grovfilen, svarende til at finde et polynomium der med meget uklare briller skærer i det aktuelle punkt. Beregningen gennemgået herunder

I artiklen ser beregningen således ud:
$x = 0.64278761, s = 106000$
$\pi = 3.141592654, B = 10.$
Så angiver de $\cos^{-1}(x) = \frac{5}{18}\pi$ og dermed $T_s(x) = \cos(106000 * \frac{5}{18}\pi)$ =0.173648178
De anfører altså Alices offentlige nøgle til at være:
$(x, T_s(x)) = (0.64278761, 0.173648178)$
I forbindelse med kryptering vælges Bob nøgle til at være r=81500.
Det fører til følgende beregninger:
$$T_r(x) = \cos(81500 * \frac{5}{15}\pi) = -0,939692621.$$
Her har indsneget sig en lille skrivefejl idet de naturligvis mener
$$T_r(x) = \cos(81500 * \frac{5}{18}\pi) = -0,939692621$$
Men det fører så til følgende beregning:
$$T_r(T_s(x)) = \cos(r * \cos^{-1}(T_s(x))) = \cos(81500 * \frac{4}{9}\pi) = 0.766044443. \qquad (1.10)$$
Dermed får de at $\cos^{-1}(T_r(x)) = \frac{8}{9}\pi$ og $\cos^{-1}(x) = \frac{5}{18}\pi$
Disse tal giver nogen rigtige pæne ligninger for r' s udfaldsrum de vil da nemlig være af formen:
$\{\pm 3.2 + 7.2k \mid k \in Z\}$. Herefter findes løsninger let til ligningen modulus 10, hvor kun anvendes et ciffer. De finder derfor løsninger til de modulære ligninger
$2k \equiv 8 (\mathrm{mod}\ 10),\ 2k \equiv 2 (\mathrm{mod} 10)$. Disse findes at være henholdsvis [4,9] og [1,6]

Problemet med disse beregninger synes at være at der overalt trods angivelse af Pi- værdi med 9 cifre anvendes 'exact' præcision i mellemregningerne.
Hvis jeg således beregner uden exacte Pi værdier får jeg:
$T_s(x) = \cos(106000 * \cos^{-1}(0.64278761))$ =0.1736908931 og
$T_r(x) = \cos(81500 * \cos^{-1}(0.64278761))$ = -0.939681214.
En afvigelse allerede fra 5. ciffer på begge beregninger.
Dette får meget stor betydning i den sidste beregning (1.10) da deres egen beregning
T_s(x)=0.173648178 siddestilles med $T_s(x) = \frac{4}{9}\pi$. Hvis man beregner $T_r(T_s(x))$ med henholdsvis det korrigerede tal jeg kom frem til og det kommatal de selv kom frem til og sammenholder med $\frac{4}{9}\pi$ fører det til vidt forskellige resultater.

1. Ved anvendelse af min korrigerede beregning på T_s(x)
$T_r(T_s(x)) = \cos(r * \cos^{-1}(T_s(x))) = \cos(81500 * 0.1736908931) = 0.9782991332$

2. Ved anvendelse af deres egen mellemregning
$T_r(T_s(x)) = \cos(r * \cos^{-1}(T_s(x))) = \cos(81500 * 0.173648178) = -0.853361074$

3. Ved anvendelse af meget høj præcision på Pi
$T_r(T_s(x)) = \cos(r * \cos^{-1}(T_s(x))) = \cos(81500 * \frac{4}{9}\pi) = 0.766044443$

4.Ved anvendelse af $\pi = 3.141592654$
$T_r(T_s(x)) = \cos(r * \cos^{-1}(T_s(x))) = \cos(81500 * \frac{4}{9}\pi) = 0.7660348921$

Der er altså tale om fire ***helt*** forskellige resultater afhængig af hvor der regnes med hvilken præcision. Så selv om de i artiklen anfører at regne med Pi med en specifik præcision gør de det ikke i de faktiske beregninger idet de anvender resultatet 0.766044443 og samtidig ser det ud som om de udvisker betydningen af dette ved at kun se på et betydende ciffer, L=1.

Der synes altså at være basis for at stille spørgsmålstegn ved om det nu også er så nemt at bryde systemet hvis man opererer med høj præcision. For selv om der teoretisk set er mange alternative r' der ville kunne bryde systemet, stiller implementationen i et endelig system visse begrænsninger på hvor præcist man kan ramme de aktuelle krydspunkter.

Floating point problematikken generelt

Når et system som det foreslåede skal implementeres støder man på en række praktiske problemer, der skyldes at systemet er baseret på de reelle tal og således anvender decimalbrøker frem for heltals-aritmetik. Praktisk implementation af reelle tal på digitale (diskrete) computere har problemer med numerisk ustabilitet, dvs. at det endelige resultat af en beregning med decimalbrøker ikke nødvendigvis er eksakt, selv indenfor den anvendte præcision.

Med andre ord, resultatet af operationer med decimalbrøker kan variere kraftigt på basis af små variationer i udgangspunktet.

For at illustrere problemstillingen vil jeg kort gennemgå principperne for implementation af

decimalbrøker på digitale computere.
Digitale (og dermed diskrete) computere kan ikke direkte lagre decimalbrøker med uendelig præcision, men må i stedet for nøjes med tilnærmede værdier der er pakket sammen i et endeligt antal bits.

Der er to mulige principper: 1) fixed point, der betyder at et tal har et fast antal decimaler før og efter kommaet, og 2) floating point, der betyder at det samlede antal cifre før og efter kommaet er fast, men at kommaets placering kan variere (deraf navnet floating point).

Heltal kan repræsenteres eksakt med fixed point data typer (hvor der er nul cifre efter kommaet) og beregninger med heltal er også eksakte, under forudsætning af a) alle tal (inklusive mellemregninger) er indenfor den range af tilladte værdier for den valgte datatype (f.eks. $-2^{31} <= x < 2^{31}$ for 32-bits heltal) og b) division bliver fortolket som heltalsdivision.

De samme principper gælder for decimalbrøker repræsenteret som fixed point datatyper, hvis man blot ser bort fra kommaet.

I floating point er et tal repræsenteret med et fortegnsbit, s (hvor 0 som regel er plus og 1 er minus), en heltals eksponent e, og en eksakt heltals mantisse, M >= 0. Tilsammen repræsenterer de tallet s * M * B^(e-E), hvor B er basen for nummersystemet (normalt 2) og E er en implementationsafhængig konstant.

Floating point er karakteriseret ved, at det resulterende talsystem ikke er lukket over for de fire regnearter, dvs. operationer med flydende komma foregår ved afrunding eller afskæring af cifre - også i mellemresultater. Det betyder, at selvom fejlen i enkelte operationer er ganske lille, kan fejlen blive stor ved sammensætning af flere operationer.

Man skal derfor holde tungen meget lige i munden når systemer der er afhængig af beregninger med decimalbrøker skal implementeres[10] på computere.

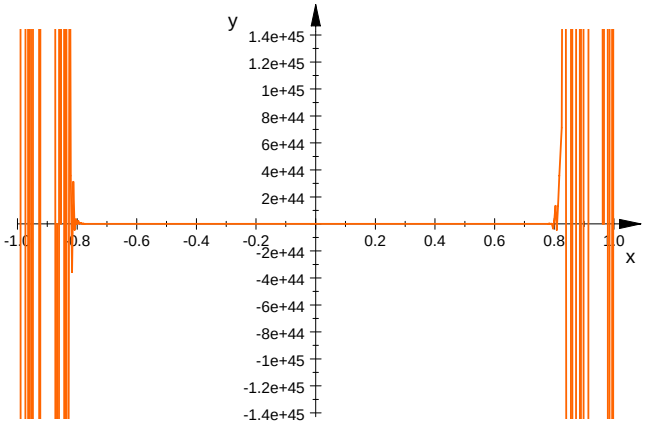
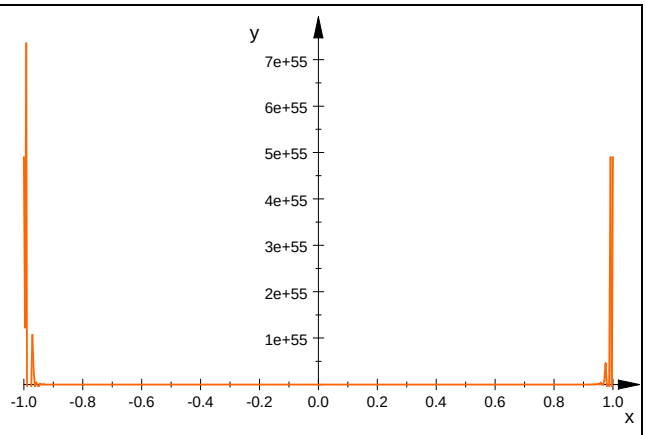
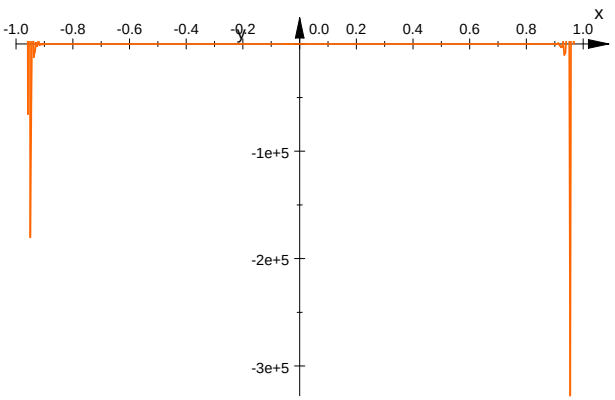
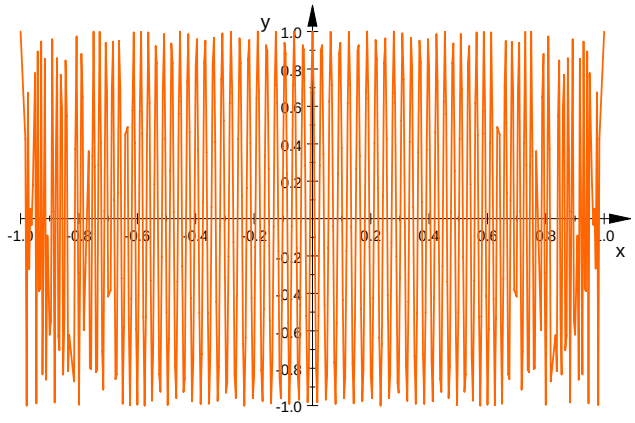
Kommatal, eller floatingpoint implementation er altså meget følsomt for afrundigsfejl og trunkeringsfejl og stiller derfor store krav til beregningspræcision.
Valget af nøgle, r og s er begrænset af antallet af cifre der regnes med i implementationen og nøglen kan således aldrig blive større end beregningspræcisionen tillader.I artiklen eksemplificeres det at en 2048 bits præcision betyder nøgler af længde mindre end 2^{970}, en grænse der er fundet ved imperi.

Samtidig er den aktuelle algoritme, Chebyshevs polynomium meget følsom over for små ændringer i præsision. For at illustrere *hvor* følsom algoritmen er over for beregningspræcision har jeg herunder illustreret chebyshevs polynomium ved en iteration på 200 og 150 med varierende præcision. Der er voldsomme udsving i resultatet udelukkende på baggrund af minimal ændring antallet af cifre der regnes med.

[10] http://www.daimi.au.dk/dProg2/noter/noter.pdf pp 37-40

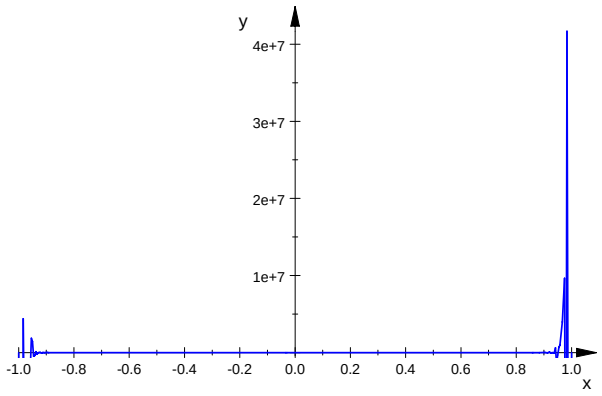
Grafer kan f.eks genereres i Mupad[11] med kommandoer af typen:
DIGITS := 66:plotfunc2d(orthpoly::chebyshev1(150, x), x=-1..1)

Chebyshevs polynomier i intervallet -1 til 1 med 200 iterationer beregnet med varierende antal cifre

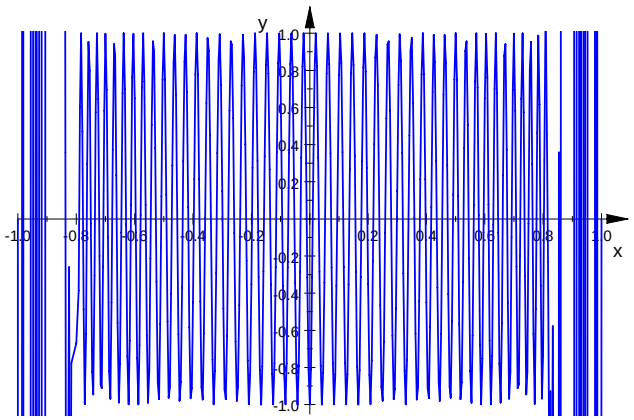
<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Her plottes chebyshevs polynomium ved 200 iterationer hvor præcisionen sættes til blot 10 cifre</td>
<td>Og dette et eksempel på same iteration med 11 cifre</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Her 65 cifres præcision</td>
<td>Billedet bliver lidt mere stabilt I intervallet f.eks ved 66 cifre.</td>
</tr>
</table>

[11] http://www.sciface.com

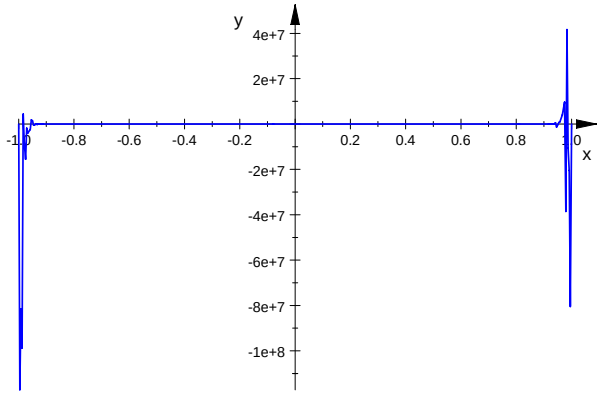
Herunder er Chebyshevs polynomium beregnet med 150 iterationer



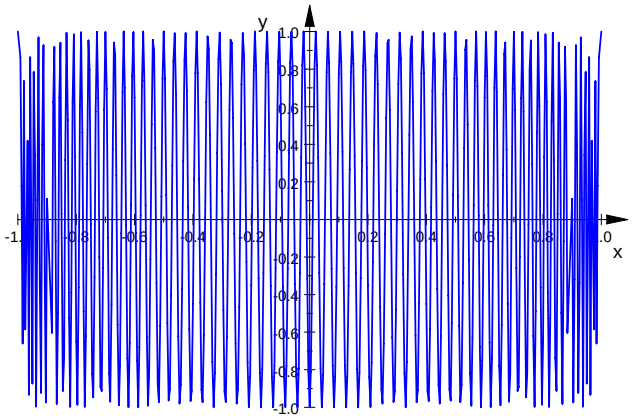
Her plottes chebyshevs polynomium ved 200 iterationer hvor præcisionen sættes til blot 40 cifre



Og dette et eksempel på same iteration med 41 cifre



Her 46 cifres præcision



Hvis vi så ønsker stor præcision f.esk 300 cifre finder vi at der er væsentlige variationer fra de 41 til de 300

Systemet ses altså på alle planer at være meget følsomt over for 'opløsning'.
Det er også væsentligt at pege på at det er vigtigt at man anvender *samme* præcision i begge ender af en kommunikation.

En variant til kryptosystemet

Jacobian Elliptic Chebyshev Rational Maps
I den den aktuelle artikel af BASK vælger man desuden også at se på et udvidet rekursivt system, baseret på Jacobi Elliptiske Chebyshev afbildninger, også ækvivalent med Jacobi elliptiske funktioner.
En elliptisk[12] funktion er en dobbelt periodisk funktion uden singulariteter – undtagen poler i det komplexe plan.
Der findes en række egenskaber[13] ved Elliptiske funktioner som gør dem interessante i krypteringsmæssig sammenhæng.
Men i dette tilfælde kan Jacobi Elliptiske Chebyshev afbildninger ses som en generalisering af Chebyshevs polynomier (eller omvendt Chebyshevs polynomin som et specialtilfælde af Jacobi Elliptiske Chebyshev afbildninger) og derfor er det interessant at undersøge om det vil give forbedringer i sikkerheden med en sådan lidt mere avanceret algoritme.

$$R_{p+1}(w,k) = \frac{2w}{1-k^2(1-R_p(w,k)^2)(1-w^2)} R_p(w,k) - R_{p-1}(w,k)$$

Altså en rekursiv definition, der så har følgende startbetingelser:
$R_0(w,k) = 1$ og $R_1(w,k) = w$

Hvis k sættes lig 0 ses det straks at alle led under brøkstregen bliver 0 på nær det første og vi får da:

$R_{p+1}(w,0) = \frac{2w}{1-0} R_p(w,0) - R_{p-1}(w,0) = 2R_p(w,0) - R_{p-1}(w,0)$, hvilket svarer helt til rekursionsformlen for Chebyshevs polynomier af første art.
Den afbildning kommuterer også under komposition og det gør den velegnet til kryptosystem:

$$R_r(R_s(w,k),k) = R_{rs}(w,k) = R_s(R_r(w,k),k) \qquad (1.11)$$

Kryptosystemet er meget lig Chebyshev Polynom systemet

1. Nøglegenerering
   Alice genererer stort heltal s sm sin hemmelige nøgle og vælger to tilfældige tal $w \in [-1,1]$ og $k \in [0,1]$ og beregner $R_s(w,k)$. (w,k, $R_s(w,k)$)udgør den offentlige nøgle
   2.Kryptering
   Bob ønsker at sende beskeden $M \in [-1,1]$ og genererer et stort heltal r.
   Han beregner derefter besteden C= $(R_r(w,k),X))$ ved at beregne $R_r(w,k), X = M / R_{rs}(w,k)$
3. Dekryptering
   Alice beregner let $M = X / R_{sr}(w,k)$ idet der jo gælder (1.11)

Angrebet bliver af BASK beskrevet på samme måde som angrebet på krypteringssystemet med Chebyshevs, idet det kan vises at det er muligt at konstruerere et r' således at $R_{r'}(w,k) = R_r(w,k)$ og dermed kan dette bruges til angrebet.

[12] http://mathworld.wolfram.com/jacobiEllipticFunctions.html
http://www.mbay.net/~cgd/flt/flt03.htm
[13] http://www.rsasecurity.com/rsalabs/node.asp?id=2012

Andre måder og perspektiv

Flere anvendelser
Også nøgleudveksling og autentifikation baseret på disse algoritmer gennemgåes i artiklen af BASK og påpeges at være sårbare over for samme type angreb som allerede beskrevet. Der er dog ikke noget der tyder på at opløsningsproblematikken er anderledes her.
Det synes altså at være centralt for disse systemer baseret på reelle tal at undersøge real-life betingelser for implementation nærmere.

Artiklen af BASK rundes af med overvejelser omkring muligheden for overhovedet at implementere systemer af denne type da de selv om de så matematisk set skulle vise sig sikre op til trapdoor altid vil være afhængig af en endelig opløsning, i en endelig implementation med endelig processorkraft og endelig hukommelse til rådighed.

En række forskellige forfattere[14] beskæftiger sig med forskellige facetter af sikkerheden i kryptering med kaotiske systemer i endelig computer præcision. F.eks påpeger Li,Mou og Cai i artiklen "Improving Security of a Chaotic Encryption Approach" hvordan forskellige ændringer i valg af værdier og afbildning kan forbedre sikkerheden. Til gengæld påpeger de også at krypteringshastigheden er ganske meget langsommere end konventionel kryptering.
Generelt ser det ud som om der er stykke vej endnu før et sikkert og effektivt kaos baseret kryptosystem er et attraktivt alternativ.

Hvis vi vender blikket den anden vej har Fee og Monagan[15] foreslået et et kryptosystem også på Chebyshevs polynomier men beregnet modulus p.
Det ser ret interessant ud og en Diffie-Hellman nøgleudveksling med Chebyshevs polynomier fungerer som følger :

I stedet for den generaliserede regel for eksponenter i en arbitrær gruppe $(g^m)^n = g^{mn} = (g^n)^m$ anvender vi kommutationsegenskaben (1.6)
1. Alice genererer positivt heltal g og primtal p så g<p
2. Alice vælger sin hellelige nøgle heltallet m så 0<m<p
3. Alice beregner $a = T_m(g) \bmod p$
4. Alice sender p,g,a til Bob

5. Bob vælger en hemmelig nøgle heltallet n så 0<n<p
6. Bob beregner $b = T_n(g) \bmod p$
7. Bob sender b til Alice

8. Alice beregner hemmelig nøgle $T_m(b) \bmod p = T_m(T_n(g)) \bmod p = T_{nm}(g) \bmod p = k$
9. Bob beregner hemmelig nøgle $T_n(a) \bmod p = T_n(T_m(g)) \bmod p = T_{nm}(g) \bmod p = k$

[14] 'Chaos and Cryptography' af Dachselt og Schwartz
'Security problems with a chaos-based deniable authentication scheme' af Gonzalo Alvarez
'On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision' af Li,Mou, Cai og Zhang
'Improving Security of a Chaotic Encryption Approach' af Li,Mou og Cai

[15]'Cryptography using Chebyshev polynomials' af Fee og Monagan

Et brud foreslås foretaget ved at løse ligningen $a = T_m(g) \bmod p$ for en ukendt grad m. Forfatterne i artiklen viser hvordan dette såkaldte Chebyshev diskrete logaritme problem kan transformeres til et konventionelt logaritmeproblem. Fee og Monagan viser ligeledes hvordan en RSA algoritme kan konstrueres på Chebyshev polynomier.

Der synes altså at være en række muligheder for at slippe for den lidet attraktive implementation med reelle tal og alligevel basere et kryptosystem på en kaotisk algoritme. Konklusioner vil dog kræve nærmere studier af kryptering med kaos i modulære rum og er da også næste skridt for nærværende forfatter. ¤