5 Chiffer med öppen nyckel

B. Franklin. Om alla uttryckte sig rakt på sak skulle det aldrig bli någon konversation.

I detta kapitel beskrivs några system med öppen nyckel. Här antas att chiffren används för konfidentialitet. Digitala signaturer för integritet och dataursprungsautenticitet behandlas i nästa kapitel.

En del metoder förutsätter att klartexten blockindelas. Ett vanligt sätt att förfara är då att använda ECB, där blocken chiffreras oberoende av varandra. Ett alternativ vore att kedja chifferblock; CBC. Detta för att förhindra att block obemärkt omordnas. I praktiken används dock PKS oftast för att kryptera korta meddelanden t ex nycklar, tidsangivelser, slumptal eller identitetsinformation.

Både OFB- och CFB-modus är dock oanvändbara som PKS eftersom chiffrering och dechiffrering där sker med samma nyckel.

5.1 RSA

5.1.1 Systemet

RSA-algoritmen bygger på att *e* är "lätt" att beräkna men att *d* i praktiken låter sig beräknas endast via tillgång till extra information (kallas ibland fall-lucke-parameter); d v s den privata nyckeln.

Metoden består av ett blockchiffer som bygger på exponentiering (modulo *n*) och svårigheterna att faktorisera (stora) primtal. Ett PKS kan aldrig ge perfekt sekretess (entydighetslängden är 0 (exakt!)), så strävan är efter beräkningsmässig sekretess.

Konstruktören utgår från två stora, t ex 200-siffriga, primtal *p* och *q* och bildar produkten $n = p * q$. Faktorerna i *n* hålls hemliga medan *n* kan ges ut publikt. Att återvinna *p* och *q* kräver att det är möjligt att primtalsfaktorisera stora och välvalda tal. Ingen har ännu konstruerat en tillräckligt snabb metod för detta ändamål.

Snabbare datorer duger inte heller eftersom svårigheten att faktorisera växer exponentiellt medan svårigheten att exponentiera och testa för primtal växer polynomiellt med antalet bitar i argumenten.

i. Specifikation. RSA specificeras enligt nedanstående tabell.

Användaren definierar ett RSA-system enligt följande.

1. Generera två stora primtal *p* och *q*.
2. Beräkna $n = p\,q$ och $\phi(n) = (p - 1)\,(q - 1)$.
3. Välj ett $e \in [1, \phi(n) - 1]$ sådant att $gcd\,(e, \phi(n)) = 1$.
4. Beräkna $d = e^{-1} \bmod \phi(n)$ med Euklides algoritm.
5. Publicera den öppna nyckeln $<n, e>$.

Alla kan nu använda chiffreringen med den öppna nyckeln *e*, men bara systemkonstruktören kan dechiffrera med den privata nyckeln *d*.

Paret $<x, e>$ bör konstrueras så att e_K innebär en modulo-reduktion, annars kan man "lätt" beräkna *e*:te roten. Den vanliga tekniken består i att "salta" korta meddelanden med redundans.

Parametrar	**Alfabet**
Talet $n = p\ q$, där p och q är primtal sådant att faktorisering av n är beräkningsmässigt ogenomförbart	$\mathtt{M} = \mathtt{C} = Z_n$ $\mathtt{K} = \{<p, q, n, e, d>:$ $e\ d \equiv 1\ (mod\ (\phi(n))\}$ Publik nyckel: n, e Privat nyckel: d, p, q
Chiffrering	**Dechiffrering**
$y = e_K(x) = x^e\ mod\ n$ $x \in \mathtt{M}$ $y \in \mathtt{C}$ $K \in \mathtt{K}$	$x = d_K(y) = y^d\ mod\ n$ $x \in \mathtt{M}$ $y \in \mathtt{C}$ $K \in \mathtt{K}$

Tabell 5.1. RSA

ii. Not. RSA-algoritmen är betydligt mer (cirka 1000 gånger) beräkningskrävande än klassiska en-nyckelsystem som DES. Detta gör att PKS i praktiken endast används vid tillämpningar som behövs sällan:

- nyckelhantering: kommunikation av krypteringsnycklar (för konventionella chiffer).
- autenticering: lösenord kan krypteras, identifikationsprotokoll kan användas.
- för digitala signaturer.

Konventionella kryptosystem används däremot för:

- integritet (meddelandeautenticitet): sk kryptografiska checksummor.
- konfidentialitet för information som utbyts mellan 'peer entities'.

Observera allmänt att kryptering kan användas både för data som lagras i t ex filsystem och data som sänds över kommunikationskanaler som kan avlyssnas.

Däremot är det tyvärr oftast omöjligt att bearbeta data i krypterad form för att först efter en slutdekryptering ta fram resultatet; tex gäller allmänt att

$$d\ (e(x)\ \blacklozenge\ e(y)) \neq x\ \blacklozenge\ y,$$

där ♦ står för en aritmetisk/logisk operator.

iii. Huvudsats (för RSA).

Om $n = p\ q$, $gcd(e, \phi(n)) = 1$ och $e\ d\ mod\ \phi(n) \equiv 1$ så gäller för paret

$$y = e(x) = x^e\ mod\ n \text{ och } d(y) = y^d\ mod\ n$$

att $d(e(x)) = x^{ed}\ mod\ n = x$, för alla $x \in Z_n$.

Bevis.

Kongruensen $e\ d\ mod\ \phi(n) \equiv 1$ innebär att $e\ d = k(p - 1)\ (q - 1) + 1$ för någon konstant k.

Fermats sats säger att $x^{ed} \equiv (x^{(p-1)})^{k(q-1)}x \equiv 1^{k(q-1)}x \equiv x\ (mod\ p)$ om x mod $p \neq 0$.

Om $x \equiv 0$ (*mod p*) så gäller uppenbarligen också $x^{ed} \equiv x$ (*mod p*).

Fermats sats säger att $x^{ed} \equiv (x^{(q-1)})^{(p-1)}x \equiv 1^{k(p-1)}x \equiv x$ (*mod q*) om $x \bmod q \neq 0$.

Om $x \equiv 0$ (*mod q*) så gäller uppenbarligen också $x^{ed} \equiv x$ (*mod q*).

Enligt CRT gäller då att $x^{ed} \equiv x$ (*mod pq*) för alla $x \in Z_n$, d v s att $d(e(x)) = x$.

5.1.2 Kommentarer

i. Lite speciellare. Många framställningar visar satsen bara för specialfallet $x \in Z_n^*$. Det kan man göra med Eulers sats och utan att referera till CRT:

Om $e\ d \bmod \phi(n) = 1$, så finns det ett k sådant att $e\ d = k\phi(n) + 1$. Om $x \in Z_n^*$ så gäller

$$d_k\ e_k(x) \equiv x^{ed} \equiv x^{k\phi(n)+1} \equiv (x^{\phi(n)})^k\ x \equiv 1\ x\ (mod\ n).$$

Detta är i skenet av ii. pragmatikerns väg.

ii. Om gcd (x, n). Sannolikheten för att ett meddelande x inte är prima relativt n är

$$p(gcd\ (x, n) > 1) = 1 / p + 1 / q - 1/ p\ q,$$

vilket är mindre än 10^{-99} om p och q är 100-siffriga primtal:

I intervallet [1, n] finns det $\phi(n)$ tal som är coprima n och alltså $n - \phi(n)$ som inte är det.

Sannolikheten ovan blir alltså

$$(n - \phi(n)) /n = [p\ q - (p-1)\ (q-1)] / p\ q\ = 1/p + 1/q - 1/pq \approx 1/p + 1/q.$$

Vidare gäller att om en kryptoanalytiker upptäcker att $gcd(x, n) > 1$, så kan n enkelt faktoriseras, varför parametrarna ändå inte duger:

Eftersom $x < n$ så är $gcd(x, n) = p$ eller q.

iii. Euklides algoritm kan användas för att lösa en ekvation av typ

$$e\ d \equiv 1\ mod\ (\phi(n))$$

då $gcd(e,\phi(n)) = 1$.

Eftersom $\phi(n)$ är jämnt så följer att e och d måste vara udda.

iv. Faktorisering. Om $\phi(n)$ avslöjas så låter sig p och q härledas och därmed också d varvid chiffret är forcerat.

Ty om n och $\phi(n)$ är publika är det lätt att faktorisera $n = p\ q$: Systemet

$$n = p\ q$$

$$\phi(n) = (p- 1)\ (q - 1)$$

ger efter elimination av q andragradsekvationen

$$p^2 - (n - \phi(n)\ + 1)p + n = 0,$$

som har två lösningar som är faktorerna p och q i n.

Det är alltså inte lättare att bestämma $\phi(n)$ än att faktorisera n.

Om å andra sidan en motspelare på något sätt kan bestämma *d* så låter sig *n* faktoriseras som följer.

Med $ed - 1 = k\phi(n) = 2^s t$ kan man visa att $a^x \equiv 1 \ (mod\ n)$, där $x = 2^i t$, för något $i \in [1, s]$ för minst hälften av alla $a \in Z_n^*$ och att $gcd(n, a^x)$ är en icke-trivial faktor av *n*.

Det betyder att i genomsnitt två sådana *a* behöver provas för att faktorisera *n*.

v. Om exponenten. De flesta presentationer använder relationen $e\ d \equiv 1 \ (mod\ \phi(n))$ som grund för RSA.

I själva verket fungerar även villkoret

$$e\ d \equiv 1 \ (mod\ \gamma(n)), \qquad (*)$$

där $\gamma(n) = lcm\ (p - 1, q - 1) = (p - 1)\ (q - 1) / gcd(p - 1, q - 1)$.

Detta leder dock sällan till en mindre exponent och effektivare beräkning eftersom $gcd(p - 1, q - 1)$ är liten om *p* och *q* väljs slumpmässigt varvid γ och ϕ blir av samma storleksordning.

Exempel. Låt $p = 13$ och $q = 17$. Då är $n = 221$, $\phi(n) = 192$ och $\gamma(n) = 48$. Om vi väljer $e = 37$ så erhålls med $e\ d\ mod\ \phi(n) = 1$ att $d = 109$, medan utnyttjande av (*) ger $d = 13$.

Giltigheten kan demonstreras som följer.

Om *p* är ett primtal så gäller enligt Fermat för alla $x \in Z_p$ att $x^{p-1} \equiv 1 \ (mod\ p)$.

Enligt CRT kan alla $x \in Z_n$ representeras entydigt som

$$x\ mod\ n \leftrightarrow \langle x\ mod\ p, x\ mod\ q \rangle$$

och $x^\alpha \equiv 1 \ (mod\ n)$ precis då $x^\alpha \equiv 1 \ (mod\ p)$ och $x^\alpha \equiv 1 \ (mod\ q)$.

Av Fermats sats följer att

$$\alpha = lcm\ (p - 1, q - 1).$$

vi. Gemensam modulus. (Ett protokollfel) Av Bezouts identitet följer att om ε och ζ är relativt prima så finns tal *r* och *s* så att

$$r\varepsilon + s\zeta = 1.$$

Detta kan användas för att påvisa en svaghet med RSA: Problemet med gemensamt modulovärde; ε och ζ är de publika nycklarna för två chiffreringar.

Om $c_1 = m^\varepsilon\ mod\ n$ och $c_2 = m^\zeta\ mod\ n$, så kan en forcör som uppsnappar c_1 och c_2 bilda

$$c_1^r c_2^s \equiv m^{r\varepsilon + s\zeta} \equiv m \ (mod\ n),$$

om $gcd(\varepsilon, \zeta) = 1$ och $r\varepsilon + s\zeta = 1$, och därigenom bestämma *m* utan att forcera RSA i sig.

Observera att ett av talen *r* eller *s* är negativt och att en exponentiering med ett negativt tal innebär att inversen måste beräknas. Om $r < 0$ så betyder $m^r = (m^{|r|})^{-1}$.

Sense moral: Använd olika *n* i olika chiffersystem.

vii. Beräkningstid. En liten lustig tillfällighet är att det är lika "svårt" att beräkna den diskreta logaritmen som att faktorisera primtal; nämligen (bara LITE mindre än)

Ordo($e^{sqrt(ln\,(n)\,ln\,(ln\,(n))}$) < Ordo ($n$),

vilket betyder att svårigheten fördubblas varje gång man utökar n med en bit.

viii. Fixpunkter. RSA har en egenhet att ibland inte dölja m; d v s att

$c = m^e \bmod n = m$.

ix. Sats. Om RSA med parametrar $n = p\ q$ och e väljes så finns det α stycken m som chiffreras till m, $m = e(m)$ Ett sådant m kallas en fixpunkt för $e()$. Det gäller att

$\alpha = [1 + gcd(e - 1, p - 1)] * [1 + gcd(e - 1, q - 1)]$.

Bevis.

Ett meddelande m är odöljbart precis då $m^e \equiv m\ (mod\ n)$. Denna kongruens är ekvivalent (enligt CRT) med paret

$m^e \equiv m\ (mod\ p)$ (1)
$m^e \equiv m\ (mod\ q)$ (2)

Ekvationen (1) kan skrivas som

$m^{e-1} \equiv 1\ (mod\ p)$ (3) eller

$m^{e-1} \equiv 0\ (mod\ p)$

och analogt för (2).

Ekvationen (3) har $gcd(e - 1, p - 1)$ lösningar och därmed följer satsen. Det visas som följer.

Låt α vara ett primitivt element i Z_p^*. Det betyder att varje m kan skrivas $m = \alpha^u$ för något entydigt givet u.

Ekvation (3) ger då

$\alpha^{u(e-1)} \equiv 1\ (mod\ p)$.

Men eftersom α är ett primitivt element gäller för x sådana att $\alpha^x \equiv 1\ (mod\ p)$ att

$x \equiv p - 1\ (mod\ (p - 1))$.

Här är alltså

$(e - 1)\ u \bmod (p - 1) = p - 1$. -- jfr $ax\ mod\ n = b$.

Denna ekvation har $d = gcd\ (a, n)$ d v s $d = gcd\ (e - 1, p - 1)$ lösningar då $d \mid b$, vilket är uppenbart i detta fall.

Exponenten e bör alltså väljas så att $gcd\ (e - 1, p - 1)$ är liten.

x. Itererad attack. Ett annat tänkbart sätt att forcera RSA vore att givet $c_0 = m^e\ mod\ n$ pröva att generera

$c_i = c_{i-1}^e\ mod\ n$

till dess $c_i = c_0$.

I detta läge gäller att $c_{i-1} = m$. Sannolikheten för att detta ska lyckas är extremt liten om primtalen (p) väljs så p - 1 har en stor primfaktor p' och p' - 1 också har en stor primfaktor p''.

Detta betyder att antalet iterationer som behövs är $ord_{\phi(n)}(e)$. Alltså bör denna ordning vara stor.

Med bra parametrar är denna typ av attack verkningslös och man kan visa att om attacken lyckas så är det lätt att faktorisera n.

xi. Små exponenter. Om tre användare har samma och en liten publik exponent $e = 3$ men olika modulus n_1, n_2 respektive n_3 och någon vill sända samma meddelande x till alla tre så överförs alltså

$$y_i = x^3 \bmod n_i.$$

En forcör kan genom att observera $\{y_i\}$ härleda x utan att faktorisera n_i. Antag att $gcd(n_i, n_j) = 1$, annars kan "gemensam modulus" tillämpas.

Ett $y \in Z_n$ sådant att $y = x^3 \bmod (n_1 n_2 n_3)$ kan bestämmas mha CRT.

Eftersom $x < min(n_i)$ så är $x^3 < n_1 n_2 n_3$, vilket betyder att

$$x^3 = y.$$

Att ta tredje roten ur ett heltal är enkelt!

Ett sätt att komma tillrätta med detta är att genom att tillfoga en slumpsekvens ("att salta") till varje klartext som chiffreras tillse att samma x aldrig uppkommer i praktiken.

Att välja en liten privat exponent d är inte heller lyckat: Om $gcd(p - 1, q - 1)$ är liten, vilket är det vanliga fallet, så kan små d-värden härledas lätt från den publika nyckeln.

xii. RSA med CRT. Vid beräkning av $m = c^d \bmod n$ kan CRT användas för att få en effektivare beräkning än vad en naiv tillämpning av *fastexp (c, d, n)* ger.

I fallet med två primtal p och q, $p < q$, gäller att ekvationssystemets

$$x \equiv a \pmod p$$

$$x \equiv b \pmod q$$

lösning enligt CRT kan skrivas

$$x = ((a - (b \bmod p)\, u) \bmod p)\, q + b \qquad \text{om } a \geq b$$

$$x = ((a + p - (b \bmod p)\, u) \bmod p)\, q + b \qquad \text{om } a < b$$

där $u\, q \equiv 1 \pmod p$.

Det är trivialt att $x \equiv b \pmod q$. Följande uträkningar visar att ($x \equiv a$)

$$x \equiv (a - b)\, u\, q + b \equiv (a - b) + b \equiv a. \qquad \text{alla led } (\bmod p)$$

Entydighten är visad i kapitel 4.

Överfört på RSA gäller det alltså att bestämma $x = m$ = klartexten då

$$a = c^d \bmod p$$
$$b = c^d \bmod q.$$

Med beteckningarna

$d = k\,(p - 1) + r = j\,(q - 1) + s,$

dvs att $d \equiv r\ (mod\ (p - 1))$ och $d \equiv s\ (mod\ (q - 1))$,

så kan a och b förenklas med hjälp av Fermats sats på följande sätt

$a = (c^{p-1} \bmod p)^k c^r \bmod p = (c \bmod p)^r \bmod p$

$b = (c^{q-1} \bmod q)^j c^s \bmod q = (c \bmod q)^s \bmod q.$

Dechiffreringen kan nu beräknas på följande sätt:

1. Beräkna

   $a = (c \bmod p)^{d \bmod (p-1)} \bmod p$

   $b = (c \bmod q)^{d \bmod (q-1)} \bmod q.$

2. Lös ekvationen $u\,q \equiv 1\ \ (mod\ p)$.

3. Erhåll klartexten enligt någon av formlerna

   $m = (((a - (b \bmod p))\ u \bmod p)\ q + b$ om $a \geq b$

   $m = (((a + p - (b \bmod p))\ u \bmod p)\ q + b$ om $a < b$.

Detta är (nära nog) den optimala beräkningen.

Not. Ovanstående har beskrivits utgående från dekrypteringsfunktionen eftersom d-värdet "blir vad det blir" (stort och besvärligt) medan e-värdet ofta väljs så att många bitar är 0, tex e = 100 ... 001. Dessutom krävs ju tillgång till p och q. Antlaet 1-bitar kallas Hammingvikten. En exponent med låg Hammingvikt ger en snabb exponentiering.

xiii. Meddelandekodning. En klartext kan kodas (trivialt) genom att ASCII-värdena för symbolerna konkateneras. Ett alternativ framgår av följande exempel. Antag att $\mathtt{M} = Z_{26} = \{\mathtt{a}, ..., \mathtt{z}\} = \{0, ..., 25\}$.

Utgå från det givna modulovärdet n och betrakta ett klartextblock av längd t, där t är det största värdet sådant att

$25 * 26^t + 25 * 26^{t-1} + ... + 25 * 26 + 25 \leq n.$

Underförstått är här 25 det största värdet av "baskoden" för någon bokstav nämligen z.

Exempel. Med n = 18932 inses att t = 2 varför exempelvis följande koder erhålls.

`aaa`	↔	0 =	0
`cat`	↔	$2 * 26^2 + 0 * 26 + 19$ =	1371
`zzz`	↔	$25 * 26^2 + 25 * 26 + 25$ =	17575

Processen måste förstås inverteras vid dekryptering. Detta kan göras entydigt.

Analogt kan förfaras för att utvidga e_K och d_K från Z_n till Z_∞. En godtyckligt lång klartext (ett heltal) N skrivs i basen n som

$N = c_0 + c_1 n + c_2 n^2 + ... + c_k n^k + ...\ ,$

varvid e_K utvidgas via följande och analogt för d_K. (Bevisa gärna detaljerna.)

$$e_K : N \rightarrow e_K(N) = e_K(c_0) + e_K(c_1)n + ... \; e_K(c_k)n^k + ...$$

xiv. Ytterligare en attack. Om $e = 3$ och två meddelanden m och $m + 1$ krypteras så erhålls chiffren (RSA är alltså inte 'non-mallable'; se avsnitt 5.5.3.)

$$c_1 = m^3$$

$$c_2 = (m + 1)^3 = m^3 + 3m^2 + 3m + 1.$$

Då gäller att

$$m = [c_2 + 2c_1 - 1] / [c_2 - c_1 + 2] !$$

Denna attack kan generaliseras till m och $am + b$, där a och b är kända och till exponenter större än 3. Attacken kräver Ordo(e^2) operationer.

xv. Fixt e. Det är vanligt att i RSA-system använda en fix exponent e, tex $e = 2^{16} + 1 = 65537$; ett tal med bara två 1-bitar i binärrepresentationen. Detta gör att chiffreringen kan beräknas mycket snabbt.

xvi. Primtalen p och q. Talen måste väljas så att faktorisering är svår. En restriktion är att de bör vara om minst 1024 bitar vardera. Vidare bör skillnaden $p - q$ inte vara alltför liten, ty i så fall är $p \approx q$ vilket betyder att $p \approx \sqrt{n}$ och n kan faktoriseras genom att provdividera med tal nära $\sqrt{n}$. Många rekommenderar dessutom att som p (och q) välja starka primtal:

- $p - 1$ har en stor primtalsfaktor r.

- $p + 1$ har en stor primtalsfaktor.

- $r - 1$ har en stor primtalsfaktor.

xvii. 'Forward search attack'. Om definitionsområdet är litet eller förutsägbart är det enkelt att forcera ett chiffer c genom att för alla klartexter m beräkna $m^e \; mod \; n$ till dess resultatet blir c. En väg att förhindra detta består i att "salta" m.

5.2 ElGamals metod

5.2.1 Grundversionen över Z_p

Medan RSA bygger på svårigheten att faktorisera stora tal så bygger föreliggande metod på svårigheten att forcera genom att beräkna den diskreta logaritmen. Detta problem kan som tidigare sagts framställas som:

> Låt p vara ett primtal, α ett primitivt element i Z_p^* och låt $\beta \in Z_p^*$.
> Sök det entydiga talet $a \in [0, p - 2]$ sådant att $\alpha^a \equiv \beta \; (mod \; p)$.
> Beteckningen $a = log_\alpha \beta$ används för lösningen.

Om p är stort (fler än 150 siffror) och $p - 1$ har stora primfaktorer så är problemet ansett vara svårt. Den minnesgode erinrar sig dock att inversen exponentiering är enkel.

i. Definition. ElGamals PKS fördubblar chiffrets längd via en extra parameter k som väljs av avsändaren på nytt för varje chiffer och hålls privat.

Klartexten x så att säga maskeras genom multiplikation med β^k för att ge y_2. Mottagaren som vet sin privata nyckel/exponent a kan beräkna β^k från y_1 och dividera bort denna faktor från y_2 för att erhålla klartexten x.

Parametrar	**Alfabet**
Primtalet p som gör *log*-problemet svårt. Ett primitivt element $\alpha \in Z_p^*$	$M = Z_p^*$ $C = Z_p^* \times Z_p^*$ $K = \{<p, a, \alpha, \beta>:$ $\beta \equiv \alpha^a \pmod p\}$ Publik nyckel: p, α, β Extra parameter: $k \in Z_{p-1}$ Privat nyckel: a
Chiffrering	**Dechiffrering**
$e_K(x, k) = <y_1, y_2>$ $y_1 = \alpha^k \bmod p$ $y_2 = x\beta^k \bmod p$	$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$ $<y_1, y_2> \in Z_p^* \times Z_p^*$

Tabell 5.2. ElGamals PKS

ii. Bevis.

$$y_2(y_1)^{-a} \equiv x\beta^k(\alpha^k)^{-a} \equiv x(\alpha^a)^k\,(\alpha^k)^{-a} \equiv x\alpha^{ak-ak} \equiv x \pmod p.$$

Observera beteckningssättet $\alpha^{-a} \equiv (\alpha^{-1})^a \equiv (\alpha^a)^{-1}$.

iii. Anmärkningar.

1. Metoden är "skojig" i sig, men den kommer till sin rätta fördel vid framställning av digitala signaturer.

2. I stället för att bilda $y_2 = x\beta^k \bmod p$ med en vanlig multiplikation, kan metoden generaliseras genom att använda ett valfritt symmetriskt chiffer; $y_2 = e_K(x)$, $K = \beta^k \bmod p$.

3. Det är vitalt att inte samma k används mer än en gång för chiffrering. Om så skulle ske erhålls chifferdelarna

$$y_2' = x_1\beta^k \bmod p$$

$$y_2'' = x_2\beta^k \bmod p,$$

varvid $y_2' / y_2'' = x_1 / x_2$ och en känd-klartext-attack är enkel: Om x_1 är känd så kan x_2 enkelt beräknas.

5.2.2 Med elliptiska kurvor

Det är möjligt att använda andra strukturer än Z_p som underlag för att bilda additativa abelska grupper i vilka t ex den diskreta logaritmen är svår.

Ett exempel på detta utgörs av de s k elliptiska kurvorna.

Fördelen blir att betydligt mindre värden p kan användas.

i. Definition. En elliptisk kurva $y^2 = x^3 + ax + b$ över Z_p består av lösningarna

$$<x, y> \in Z_p \times Z_p$$

till

$$y^2 \equiv x^3 + ax + b \ (mod\ p), \qquad (1)$$

där p är ett primtal, $p > 3$, och $a, b \in Z_p$ sådana att

$$\Delta = 4a^3 + 27b^2 \ mod\ p \neq 0. \qquad (2)$$

Vidare ingår i den elliptiska kurvan också oändlighetspunkten (nordpolen) N.

Hokus Pokus (?): Nedan (vii.) kommer ett försök till förklaring av tex (2) via resonemang från analytisk geometri a` la Descartes (Cartesius).

ii. Exempel. Låt E vara definierad av (här är $\Delta = 8$)

$$y^2 = x^3 + x + 6 \qquad (3)$$

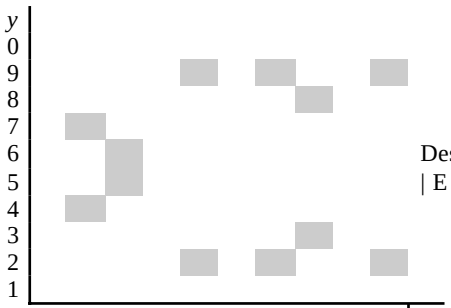
över Z_{11}. Punkterna på E kan bestämmas genom att låta x genomlöpa hela Z_{11} och sedan lösa (3) för alla x.

Med Eulers kriterium avgörs vilka $z = x^3 + x + 6$ som är kvadratiska residuer.

Eftersom talet 11 uppfyller $11 \equiv 3 \pmod 4$ så kan kvadraträtterna till de kvadratiska residuerna bestämmas explicit som

$$\pm z^{(11+1)/4} \ mod\ 11 = \pm z^3 \ mod\ 11.$$

Detta kan sammanfattas i följande figur/tabell.



Dessutom ingår N; alltså totalt 13 punkter:
| E | = 13. Observera symmetrin.

0	1	2	3	4	5	6	7	8	9	0	x
6	8	5	3	8	4	8	4	9	7	4	$z = x^3 + x + 6$ (*mod* 11)
n	*n*	*j*	*j*	*n*	*j*	*n*	*j*	*j*	*n*	*j*	$\in$ QR(11) ?
		4	5		2		2	3		2	y
		7	6		9		9	8		9	

Allmänt kan visas att en elliptisk kurva har ungefär p punkter.

Mer explicit gäller (teorem av Hasse):

$$p + 1 - 2\sqrt{p} \leq | E | \leq p + 1 + 2\sqrt{p}.$$

iii. Gruppstruktur. För att forma en grupp med hjälp av E behöver vi definiera en operation (addition).

Följande visar hur. All aritmetik utförs i Z_p.

Låt P = $<x_1, y_1>$, Q = $<x_2, y_2> \in E$.

- Om $x_2 = x_1$ och $y_2 = - y_1$ så är P + Q = N.

- Annars är P + Q = $<x_3, y_3>$,

där

$$x_3 = \lambda^2 - x_1 - x_2$$

och

$$y_3 = \lambda(x_1 - x_3) - y_1.$$

Kvantiteten λ är

$$\lambda = (y_2 - y_1) / (x_2 - x_1) \qquad \text{om } P \neq Q$$

$$\lambda = (3x_1^2 + a) / 2y_1 \qquad \text{om } P = Q.$$

- Till sist definieras för alla P

$$P + N = N + P = P.$$

Utrustad med denna operation (+) kan vi visa att *E* blir en abelsk grupp med enhetselement N. Bevisen är lite omständliga men "raka". Att bevisa associativitet är dock svårt.

Det är enkelt att beräkna inverser: Inversen till $<x, y>$, som skrivs - $<x, y>$, är $<x, - y>$.

iv. Exempel. Betrakta åter ekvationen (3). *E* har 13 punkter. Eftersom varje grupp med ordning lika med ett primtal är cyklisk så är *E* isomorf med Z_{13}.

Utgå från det primitiva elementet $\alpha = <2, 7>$. Beräkna "potenserna" av α. Dessa skrivs som multiplar eftersom *E* är additiv. För att beräkna

$$2\alpha = <2, 7> + <2, 7>$$

bestäms först λ:

$$\lambda = (3 * 2^2 + 1) / (2 * 7) = mod\ 11 = 8.$$

Alltså är $2\alpha = <x_3, y_3>$, där

$$x_3 = 8^2 - 2 - 2\ mod\ 11 = 5$$

och

$$y_3 = 8(2 - 5) - 7\ mod\ 11 = 2.$$

Därmed gäller att $2\alpha = <5, 2>$.

Genom analoga uträkningar erhålls följande:

α	2α	3α	4α	5α	6α	7α	8α	9α	10α	11α	12α
<2,7>	<5,2>	<8,3>	<10,2>	<3,6>	<7,9>	<7,2>	3,5>	<10,9>	<8,8>	<5,9>	<2,4>

Elementet α =<2, 7> är alltså verkligen en generator för given elliptisk kurva.

v. Huvudsats. (Utan bevis)

Om E är en elliptisk kurva över Z_p, $p > 3$ och primtal, så finns det tal m och n sådana att följande gäller:

$n \mid m$
$n \mid (p - 1)$
E är isomorf med $Z_m \times Z_n$.

Härav kan några slutsatser dras.

1. Om m och n är kända eller kan beräknas så har E en cyklisk delgrupp som är isomorf med Z_m. Denna kan användas t ex för ett ElGamal-chiffer.

2. Om $n = 1$ så är E en cyklisk grupp.

3. Om $| E |$ är ett primtal, eller en produkt av två primtal, så är E en cyklisk grupp.

vi. ElGamal-chiffer. Utgå från kurvan (3) och chiffrera på följande sätt.

- Låt $\alpha = <2, 7>$ vara generatorn.
- Låt Bs privata "exponent" vara $a = 7$.
- Då gäller att $\beta = 7\alpha = <7, 2>$ är den publika nyckeln.

Notera att detta ger upphov till ett problem som kan formuleras som att lösa den diskreta logaritmen över en elliptisk kurva:

Givet β, finn det a sådant att $\beta \equiv a\alpha$.

Chiffreringen blir

$$e_K(x, k) = <y_1, y_2> = <k<2, 7>, x + k<7, 2>>,$$

där k, $0 \le k \le 12$, är engångsparametern och meddelandet $x \in E$.

Dechiffreringen blir

$$d_K(y_1, y_2) = y_2 - 7y_1.$$

Med $k = 3$ och $x = <10, 9>$ så erhålls chiffret $y = <<8, 3>, <10, 2>>$ eftersom

$$y_1 = 3<2, 7> = <8, 3>$$

$$y_2 = <10, 9> + 3<7, 2> = <10, 9> + <3, 5> = <10, 2>.$$

Klartexten återvinns enligt

$$x = <10, 2> - 7<8, 3> = <10, 2> - <3, 5> = <10, 9> + <3, 6> = <10, 9>.$$

Notera att chiffrering innebär en expansionsfaktor ungefär lika med 4: Det finns cirka p möjliga klartexter och varje chiffer består av fyra element i Z_p.

Det finns ett effektivare chiffer föreslaget av Menezes och Vanstone i vilket det inte krävs att x ligger på en elliptisk kurva. Detta ger en expansionsfaktor 2.

vii. Om definitionen. Definitionen av E och gruppoperationen ser ut att vara tagna ur "tomma intet". Så är det naturligtvis inte. Här följer ett försök till att ge en informell bakgrund.

Det blir lite lättare att förstå om de reella talen R betraktas.

Tredjegradsekvationer En allmän tredjegradsekvation över R kan genom variabelbyte (visa det) skrivas på formen

$$x^3 + ax + b = 0.$$

Fyra fall kan föreligga: Ekvationen har

1. tre olika reella rötter.
2. en reell rot och två komplex-konjugerade rötter.
3. två olika reella rötter, den ena är en dubbelrot.
4. en reell trippelrot.

Fall 3 och 4 kallas singulära. En ekvation (kurva) är singulär precis då

$$\Delta = 4a^3 + 27b^2 = 0.$$

Villkoret (2) i definitionen ovan skall ses i skenet av detta förhållande.

Kurvor och punkter Antag kurvan $E = \{<x, y> \in R \times R: y^2 = x^3 + ax + b\}$ ritad i ett plan.

Givet punkterna P $= <x_1, y_1>$ och Q $= <x_2, y_2>$ med $x_1 \neq x_2$ på E. Linjen L mellan P och Q kommer L att skära kurvan E i en tredje punkt R'. Om denna punkt speglas i x-axeln erhålls en annan punkt på E som kallas R.

Definiera P + Q = R.

Den räta linjens L ekvation skrivs $y = \lambda x + \mu$ och det gäller

$$\lambda = (y_2 - y_1) / (x_2 - x_1)$$ -- riktningskoefficienten.

och

$$\mu = y_1 - \lambda x_1 = y_2 - \lambda x_2.$$ -- ordinatan vid abskissan $x = 0$.

Skärningspunkter mellan L och E söks. Substituera alltså $y = \lambda x + \mu$ i ekvationen

$$y^2 = x^3 + ax + b.$$

Detta ger efter hyfsning

$$x^3 - \lambda^2 x^2 + (a - 2\lambda\mu)x + b - \mu^2 = 0.$$

Denna ekvation har tre rötter, varav två redan är kända; nämligen x-koordinaterna x_1 och x_2 i P respektive Q.

Addition Den tredje roten erhålls då som

$$x_3 = \lambda^2 - x_1 - x_2.$$

Detta är x-koordinaten i R'. Om y-koordinaten i R' betecknas $- y_3$ så kan y_3 beräknas via lutningen λ hos L utgående från punkterna $<x_1, y_1>$ och $<x_3, - y_3>$:

$$\lambda = (- y_3 - y_1) / (x_3 - x_1)$$

eller

$$y_3 = \lambda(x_1 - x_3) - y_1.$$

Detta ger en geometrisk motivering till operationen +.

Fallet P = Q måste behandlas lite annorlunda. Definiera *L* som tangenten till *E* i punkten P.

Implicit derivering ger

$2y\ dy/dx = 3x^2 + a$.

Med $x = x_1$ och $y = y_1$ erhålls tangentens lutning

$(3x_1^2 + a) / 2y_1$.

Sedan fortsätter resonemanget i analogi med tidgare.

De är lätt att se från denna geometriska tolkning att additionen är kommutativ.

5.3 Lite om kodning

5.3.1 Allmänt

Kryptering kodar meddelanden med bibehållande av ordlängd (om man undantar de homofona, indeterministiska eller probabilistiska chiffersystem).

Idén med t ex Huffmankodning är att via redundansreduktion kompaktera meddelanden så att översändandet eller lagrandet bara behöver utnyttja "effektiva" bitar.

Vid felupptäckande eller felkorrigerande koder tillförs i stället redundans för att möjliggöra upptäckt eller korrigering av transmissionsfel; fler bitar kommer att gå åt. Denna typ av kodning kan användas för att förbättra 'integrity'.

Observera dock att kodning använt *på detta sätt* bara kan användas för oavsiktliga överföringsfel. Metoderna är helt försvarslösa mot avsiktliga manipulationer.

Dock: Med en liten 'twist' kan vissa koder användas för att bilda PKS.

Koderna implementeras i maskinvara för integritet av primärminne och skivminne inkl CD och i programvara i *i/o*-system.

Matematiska grunden för dessa koder är Galoiskropparna Z_q = GF(q) och vektorrummen $(Z_q)^n$ = GF(q)^n, n-tipler av element i GF(q), med q = 2 som ett viktigt specialfall.

i. Definitioner.

1. En kod **C** = (c_1, ..., c_m) är ett delrum av GF(q)^n, där kodorden c_j tillhör delrummet.

2. Hammingavståndet, $d(c_1,c_2)$, mellan två kodord antalet platser där kodorden är olika.

3. Minimiavståndet d^* är det minsta Hammingavståndet mellan två olika kodord.

4. Hammingvikten $wt(c)$ är lika med antalet element i c som är $\neq 0$.

5. $w^*(c)$ = (minimivikten) = minimum av w över alla $c \neq 0$.

> *För en binär (q = 2) (n,k) - kod är varje kodord en sekvens av n stycken binära symboler och det finns 2^k sådana kodord. Det innebär att en kod beskrivs av s = n*2^k bitar. Alltså finns det totalt 2^s olika koder.*
> *Exempelvis finns det för (n,k) = (40,20) cirka $10^{10\ 000\ 000}$ binära linjära blockkoder. De flesta är förstås helt värdelösa.*

Koder kan specificeras av en generatormatris **G** av dimension $k * n$ och informationsordet **i** som är en radvektor av dimension $1 * k$ genom

$$\mathbf{c} = \mathbf{iG},$$

som då blir en $1 * n$ vektor. De q^k olika kodorden kallas då en (n,k) - kod.

ii. Exempel. Informationsordet **i** = (0 1 1) och generatormatrisen

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

ger kodordet **c** = (0 1 1 1 0). Kodorden utgörs av linjärkombinationer av rader i **G**.

iii. Definition. Om **C** är en given kod med dimension k kallas det ortogonala komplementet $\mathbf{C}^{\perp}$ för den duala koden och har dimension $dim(\mathbf{C}^{\perp}) = n$ - k.

Det ortogonala komplementet $\mathbf{C}^{\perp}$ utgörs av mängden av vektorer som är ortogonala mot alla vektorer i **C**. Två vektorer **a** och **b** är ortogonala då skalärprodukten $<\mathbf{a} \mid \mathbf{b}> = \Sigma_i \, a_i b_i = 0$.

iv. Observera

1. att om **H** utgörs av basvektorer i $\mathbf{C}^{\perp}$ så gäller att $\mathbf{c}\,\mathbf{H}^{\mathrm{T}} = \mathbf{0}$.

2. matrisen **H** har dimension $(n - k) * n$ och $\mathbf{H}^{\mathrm{T}}$ är dess transponat.

3. **H** kallas 'parity check matrix'.

4. också att $\mathbf{G}\,\mathbf{H}^{\mathrm{T}} = \mathbf{0}$ (en matris av dimension $k * (n - k)$).

5. att alla **G** via kolumnpermutationer eller elementära radoperationer kan skrivas som $\mathbf{G} = (\mathbf{I} \mid \mathbf{P})$, där **I** är en enhetsmatris av dimension $k * k$ och $dim(\mathbf{P}) = k * (n - k)$.

6. att motsvarande $\mathbf{H} = (-\mathbf{P}^{\mathrm{T}} \mid \mathbf{I})$

Kodframställning enligt 4. kallas *systematisk*: Då blir $\mathbf{c} = (\mathbf{i} \mid \mathbf{x})$, där **x** kan kallas paritetsymboler eller checksumma.

v. Sats. Varje kod är ekvivalent med (kodorden är desamma med undantag av att de kan vara permuterade) en systematisk kod.

vi. Sats. Minimiavståndet för en linjär (n, k) - block-kod uppfyller $d^* \leq 1 + n - k$.

vii. Sats. Om $d^* \geq t + 1$, så kan t fel upptäckas. Om $d^* \geq 2t + 1$, så kan t fel korrigeras. Om $d^* \geq t + r + 1$, så kan t fel upptäckas och r fel korrigeras.

viii. Definition. Om det gäller att $d^* = 1 + n - k$, så kallas koden maximumavståndskod.

ix. Definition. En (n, k) block-kod säges ha (kod)hastigheten $R = k / n$.

Hamming- och Reed-Müller- koder är några av de koder som kan framställas och avkodas effektivt algoritmiskt.

5.3.2 Hammingkoder

För varje m finns en (2^m - 1, 2^m - 1 - m) - Hammingkod. Val av stora m ger kodhastigheter som är nästan lika med 1. Ett exempel får visa hur koderna byggs upp.

i. Definition. En Hammingkod är en (n, k) linjär blockkod med q, $q \geq 3$, paritetsbitar sådan att $n = 2^q - 1$ och $k = n - q$.

ii. Exempel. En (7, 4) - Hamming kod (m = 3).

Kodorden kan bildas systematiskt så att $\mathbf{c} = (i_1, i_2\ i_3\ i_4 \mid p_1\ p_2\ p_3) = (\mathbf{i} \mid \mathbf{p})$, där **i** är informationsordet och **p** är paritetsbitar.

I Hammingfallet definieras **p** genom

$$p_1 = i_1 + i_2 + i_3$$
$$p_2 = i_2 + i_3 + i_4$$
$$p_3 = i_1 + i_2 + i_4,$$

vilket också låter sig enkelt framställas av matrisformen $\mathbf{c} = \mathbf{i}\,\mathbf{G}$, där

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

Källan sänder detta kodord **c** som kan förvanskas under överföringen till

$\mathbf{v} = (i_1'\ i_2'\ i_3'\ i_4' \mid p_1'\ p_2'\ p_3') = \mathbf{c} + \mathbf{e}$, --- **e** är en felvektor

Avkodaren på mottagarsidan kan nu bilda det så kallade syndromet $\mathbf{s} = (s_1\ s_2\ s_3)$ definierat av

$$s_1 = p_1' + i_1' + i_2' + i_3'$$
$$s_2 = p_2' + i_2' + i_3' + i_4'$$
$$s_3 = p_3' + i_1' + i_2' + i_4',$$

som kan skrivas på den allmänna formen $\mathbf{s} = \mathbf{v}\,\mathbf{H}^T$, där H är "parity check matrix".

Obsevera att **s** inte beror på **i** utan bara på felmönstret.

Vidare finns åtta olika syndrom; ett av dem svarar mot "inget överföringsfel", de övriga mot vart och ett av de tänkbara enbitsfelen.

Koden kan alltså korrigera enbitsfel; $d^* = 3$ och $2t + 1 = 3$ för $t = 1$.

5.3.3 Reed - Müllerkoder

i. Allmänt. Koden användes i farkosten Mariner 9 när den 1972 skickade fotografier från Mars. Idag finns det dock bättre koder. För givna n och r med $r < n$ finns en R-M kod med blocklängd 2^m av ordning r.

Koden konstrueras på följande vis. Konstruktionen ger en kod som inte är systematisk.

Generatormatrisen framställs som $\mathbf{G} = (\mathbf{G}_0\ \mathbf{G}_1\ ...\ \mathbf{G}_r)^T$.

- Matrisen $\mathbf{G}_0$ är en $1 * 2^m$ vektor (1 1 ... 1).

- $\mathbf{G}_1$ är en $m * 2^m$- matris i vilken varje m- tipel återfinns som kolumner

0 0 0 1 1 1 1
0 1 1 0 0 1 1 (Sorterade i storleksordning; 1, ..., 7.)
1 0 1 0 1 0 1

- $\mathbf{G}_s$ konstrueras från $\mathbf{G}_1$ genom att låta raderna i $\mathbf{G}_s$ utgöras av alla "produkter" av s rader i $\mathbf{G}_1$.

"Produkter" definieras här som komponentvis multiplikation: $\mathbf{a} = (a_1 \ldots a_n)$ och $\mathbf{b} = (b_1 \ldots b_n)$ ger $\mathbf{a} \otimes \mathbf{b} = (a_1 * b_1 \ldots a_n * b_n)$. Eftersom det fins $m! / (s! (m - s)!)$ sätt att välja de s raderna i en produkt blir **G**s dimension

$$dim(\mathbf{G}_s) = m! / (s! (m - s)!) * 2^m.$$

ii. Exempel där $m = 4$, $n = 16$, $r = 3$.

$\mathbf{G}_0$ = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = (a_0)

$\mathbf{G}_1$ = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 = (a_1)
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 = (a_2)
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 = (a_3)
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 = (a_4)

Eftersom denna har 4 rader så har $\mathbf{G}_2$ 4!/ (2! (4 - 2)!) = 6 rader och $\mathbf{G}_3$ har fyra rader.

$\mathbf{G}_2$ = $a_1 \otimes a_2$ = 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
$a_1 \otimes a_3$ = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
$a_1 \otimes a4$ = 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$a_2 \otimes a_3$ = 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$a_2 \otimes a_4$ = 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$a_3 \otimes a_4$ = 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1

$\mathbf{G}_3$ = $a_1 \otimes a_2 \otimes a_3$ = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
$a_1 \otimes a_2 \otimes a_4$ = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
$a_1 \otimes a_3 \otimes a_4$ = 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
$a_2 \otimes a_3 \otimes a4$ = 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

Detta ger alltså en (16, 15) -kod och det går att bevisa att $d^* = 2^{m-r}$.

5.3.4 Cykliska koder och Galoiskroppar

i. Allmänt. Cykliska koder kan beskrivas utgående från koder över GF(q) genom att betrakta utvidgade kroppar GF(q^m).

Exempelvis kan den linjära kod (en (7, 4)-Hammingkod), med $m = 3$, som framställs av följande 'parity check matrix'

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

med element i GF(2) beskrivas enklare med element i GF(q^m) = GF(2^3) = GF(8) och med

$$H = [\alpha^0 \ \alpha^1 \ \ldots \ \alpha^7],$$

där α^i betecknar kolumner i **H** och anges med primitiva element i GF(8).

Denna kropp är konstruerad utgående från det primitiva polynomet $p(z) = z^3 + z + 1$; d v s är kvotringen (som kan göras till en kropp) över polynom modulo $p(z)$, dvs där

$\alpha^0 = 1$,
$\alpha^1 = z$,
$\alpha^2 = z^2$,
$\alpha^3 = z + 1$,
$\alpha^4 = z^2 + z$,
$\alpha^5 = z^2 + z + 1$,
$\alpha^6 = z^2 + 1$,
$\alpha^7 = 1$.

Ett primitivt polynom $p(z)$ över GF(q) är ett irreducibelt polynom sådant att i den utvidgade koppen konstruerad modulo $p(z)$ elementet z är en generator.

Observera att multiplikation och moduloreduktion sker med $p(z)$

Det finns fyra polynom av grad 3:

$z^3 + 1$,

$z^3 + z^2 + z + 1$,

$z^3 + z + 1$ och

$z^3 + z^2 + 1$.

De två första är reducibla (visa detta!) och man kan visa att de kroppar som genereras av de två senare är isomorfa. Det finns minst ett irreducibelt polynom för varje gradtal.

Relationen $\mathbf{c}\,\mathbf{H}^T = \mathbf{0}$ motsvaras alltså av $\Sigma_i\, c_i\, \alpha^i = 0$ i GF(8), varför en polynom-representation som $c(x) = \Sigma_i\, c_i\, x^i$ föreslår sig själv.

> Elementet $x = \alpha$ är ett kodord precis då $c(\alpha) = 0$.

ii. Definition. En kod kallas cyklisk om $c_{n-1}\, c_0 \ldots c_{n-2}$ är en kod då $c_0\, c_1 \ldots c_{n-1}$ är en kod.

Varje linjär kod över GF(q) av längd n är ett delrum av GF(q)^n och en cyklisk kod är ett specialfall härav. Varje vektor i GF(q)^n kan representeras av ett polynom i x av grad $\leq n - 1$, där vektorkomponenterna är lika med polynomkoefficienterna.

Polynomen utgör en (kvot)ring (betecknas ibland GF(q)/(x^n - 1)) med produkten $p(x) * p(y) = p(x)\, p(y)\ mod\ (x^n - 1)$. Ett cykliskt skift ges av multiplikation med x modulo (x^n - 1).

iii. Sats. I ringen GF(q)[x] / (x^n - 1) är en delmängd C en cyklisk kod precis då C är en delgrupp till GF(q)[x] / (x^n - 1) under addition.

Om $c(x) \in C$ och $a(x) \in$ GF(q)[x] / (x^n - 1) så gäller $a(x)\, c(x)\ mod\ (x^n - 1) \in C$.

Idén är nu att välja ett kodord ($\neq 0$) med minsta gradtal ur C, kallas $n - k$, och multiplicera detta med en skalär så att det blir moniskt: Eftersom koden är linjär blir detta också ett entydigt kodord. Detta kallas generatorpolynomet $g(x)$ för C.

iv. Sats. En cyklisk kod består av alla produkter av $g(x)$ och polynom med gradtal $\leq k - 1$.

v. Sats. Det finns en cyklisk kod med blocklängd n med generatorpolynom $g(x)$ precis då

$g(x) \mid (x^n - 1)$.

Ur detta följer att för varje cyklisk kod med generatorpolynom $g(x)$ gäller att

$$x^n - 1 = g(x)\, h(x),$$

för något $h(x)$. Detta polynom kallas "parity check polynomial". Varje kodord $c(x)$ uppfyller då $h(x)\, c(x)\ mod\ (x^n - 1) = 0$.

vi. Sammanfattning. Inför beteckningarna

$c(x)$	översänt kodord
$v(x)$	mottaget ord
$e(x)$	felpolynomet $= v(x) - c(x)$
$i(x)$	informationsordet

Kodningen kan göras (tex) på följande sätt.

1. Icke-systematiskt sätt: $c(x) = i(x)\, g(x)$; polynomet $i(x)$ "syns inte i" $c(x)$.

2. För att erhålla en systematisk kod skiftar vi in $i(x)$ i högstagradskoefficienterna, dvs

$$c(x) = x^{n-k} i(x) + t(x).$$

För att detta ska ge en kod krävs att $c(x)\ mod\ g(x) = 0$, d v s att

$$t(x) = - x^{n-k} i(x)\ mod\ g(x).$$

Observera att 1. och 2. ger samma mängd av kodord men associationen mellan i och c blir olika.

Syndrompolynomet definieras som

$$s(x) = v(x)\ mod\ g(x) = e(x)\ mod\ g(x),$$

som alltså bara beror på e och inte på c eller i.

Som stöd för minnet finns följande tabell.

Polynomnamn	*Beteckning*	*Gradtal*	*Samband/definition*
Informations-	$i(x)$	$k - 1$	Givet
Mottaget	$v(x)$	$n - 1$	$v = c + e$, givet
Generator-	$g(x)$	$n - k$	Givet
'Parity check'	$h(x)$	k	$h\, g = x^n - 1$
Fel-	$e(x)$	$n - 1$	$e = v - c$
Kodords-	$c(x)$	$n - 1$	$c = x^{n-k} i - x^{n-k} i\ mod\ g$
Syndrom	$s(x)$	$n - k$	$s = e\ mod\ g = v\ mod\ g$

Tabell 5.3. Polynom mm

För avkodning och felkorrigering kan mottagaren, givet $v(x)$, bestämma $s(x)$. Paret $\langle s,e \rangle$ kan förtabelleras och ur denna tabell kan $e(x)$ bestämmas. Snabbare algoritmer finns dock.

De användbara koderna (linjära blockkoder eller cykliska koderna) måste vara effektivt avkodbara!

vii. Kommentarer. Olika sätt att välja generatorpolynom ger olika klasser av koder. De mera omtalade är Bose- Chandhuri- Hocquenghem (BCH) - koder och specialfallet Reed-Solomon - koder. De senare används för tex CD-skivor. Goppa-koder tillhör samma klass.

I ITUs (f d CCITTs) rekommendationer för skikt 2 i "OSI-stacken" används cyklisk kod (kallas där CRC, 'cyclic redundancy check') baserad på generatorpolynomet

$$x^{16} + x^{12} + x^5 + 1.$$

Kretsar för felkoder fanns redan i de sk HDLC-'chip' som en tid (början av 70-talet) var vanliga för datakommunikationstillämpningar. Numera ingår sådana i alla 'network interfaces'.

Vidare är alla primärminnen idag försedda med felkorrigerande eller åtminstone felupptäckande koder.

5.4 McEliece-PKS

Detta PKS bygger på att dekryptering är ett enkelt specialfall av ett NP-fullständigt problem; i det här fallet avkodning av en linjär blockkod. Specialfallet här är de s k Goppa-koderna.

En Hamming-kod är ett exempel på en Goppa-kod. I detta PKS döljs alltså, för alla utom den rättmätige mottagaren, den "effektivt avkodbara" koden genom att generatormatrisen multipliceras med två privata matriser **S** och **P**, varvid problemet att avkoda chiffret blir NP-fullständigt.

Parametrarna för en Goppa-kod är $n = 2^m$, $d = 2t + 1$ och $k = n - m\,t$. Konkreta förslag är $m = 10$ och $t = 50$. Varje klartext blir då 524 bitar och varje chiffer (kod) 1024 bitar (fördubblat). Den öppna nyckeln är då en 524 * 1024-matris.

i. Metoden är som följer.

Parametrar	**Alfabet**
G en generatormatris för en (n, k, d) Goppa-kod **C**, där $n = 2^m$, $d = 2t + 1$, $k = n - m\,t$. **S** en inverterbar $k * k$-matris över Z_2 och **P** en $n * n$ permutationsmatris **G'** = **S G P**	Klartextalfabetet $\mathtt{M} = (Z_2)^k$. Chifferalfabetet $\mathtt{C} = (Z_2)^n$. Nyckelrymd $\mathtt{K}$ = {<**G**, **S**, **P**, **G'**>} **G'** är den publika nyckeln. De övriga är privata.
Chiffrering	**Dechiffrering**
$\mathbf{y} = e_k(\mathbf{x}, \mathbf{e}) = \mathbf{x}\,\mathbf{G}' + \mathbf{e}$ **e** är en slumpvektor i $(Z_2)^n$ med vikt t	Beräkna $\mathbf{y}_1 = \mathbf{y}\,\mathbf{P}^{-1}$ Avkoda $\mathbf{y}_1$ för att erhålla $\mathbf{y}_1 = \mathbf{x}_1 + \mathbf{e}_1$, med $\mathbf{x}_1 \in \mathbf{C}$ Beräkna $\mathbf{x}_0 \in (Z_2)^k$ så att $\mathbf{x}_0\,\mathbf{G} = \mathbf{x}_1$ Beräkna $\mathbf{x} = \mathbf{x}_0\,\mathbf{S}^{-1}$

Tabell 5.4. McElieces PKS

ii. Exempel Låt **G** vara (en privat) generatormatris för en (7, 4) - kod. I själva verket är detta en Hammingkod.

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

Antag vidare att följande matriser **S** och **P** väljs som privat nyckel.

$$\mathbf{S} = \begin{pmatrix} 1&1&0&1 \\ 1&0&0&1 \\ 0&1&1&1 \\ 1&1&0&0 \end{pmatrix} \qquad \mathbf{P} = \begin{pmatrix} 0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0 \end{pmatrix}$$

Den publika generatormatrisen $\mathbf{G'} = \mathbf{S\,G\,P}$ blir då

$$\mathbf{G'} = \begin{pmatrix} 1&1&1&1&0&0&0 \\ 1&1&0&0&1&0&0 \\ 1&0&0&1&1&0&1 \\ 0&1&0&1&1&1&0 \end{pmatrix}$$

Klartexten $\mathbf{x} = (1\ 1\ 0\ 1)$ chiffreras genom att använda en slumpvektor **e** med vikt 1, säg, $\mathbf{e} = (0\ 0\ 0\ 0\ 1\ 0\ 0)$ genom att räkna ut

$$\mathbf{y} = \mathbf{x}\ \mathbf{G'} + \mathbf{e}.$$

Alltså: $\mathbf{y} = (0\ 1\ 1\ 0\ 1\ 1\ 0)$.

Mottagaren (ägaren av den privata nycken) beräknar först $\mathbf{y}_1 = \mathbf{y}\ \mathbf{P}^{-1} = (1\ 0\ 0\ 0\ 1\ 1\ 1)$.

Detta avkodas till $\mathbf{x}_1 = (1\ 0\ 0\ 0\ 1\ 1\ 0)$. (Obs att $\mathbf{e}_1 \neq \mathbf{e}$.)

Värdet $\mathbf{x}_0$ är de fyra fösta komponenterna i $\mathbf{x}_1$, så att $\mathbf{x}_0 = (1\ 0\ 0\ 0)$.

Slutligen beräknas klartexten $\mathbf{x} = \mathbf{S}^{-1}\ \mathbf{x}_0 = (1\ 1\ 0\ 1)$.

5.5 Probabilistiska chiffer

Målet med probabilistisk kryptering är att ingen information om klartexten ska kunna beräknas (inom polynomiellt lång tid) givet kryptogram.

I ett PKS är det trivialt att bestämma klartexten ur ett kryptogram om klartexten bara är en bit.

En probabilistisk kryptering (PE) i stället för en deterministisk införs för att undvika att information om klartexten överförs till chiffret.

I ett PE-system svarar mot varje klartext en mängd chiffer, som alla dekrypteras till samma klartext.

$$c_i = e_k\,(m),\ i = 1, \ldots, n.$$

$$m = d_k\,(c_i),\ i = 1, \ldots, n.$$

Med detta kan en forcör inte kryptera slumptexter för att erhålla rätt klartext.

Om forcören besitter ett c_j, råkar gissa det korrekta m-värdet och beräknar $e_k(m)$ så är detta med försumbar sannolikhet i överensstämmelse med c_j.

En nackdel är förstås att detta implicerar att chiffren blir längre än klartexterna: Det är ofrånkomligt.

En metod med rimlig effektivitet är Blums och Goldwassers metod som bygger på en pseudo-slumptalsgenerator (PRBG, 'pseudo random bit generator') efter Blum, Blum och Shub; den s k BBS-generatorn.

5.5.1 BBS-generatorn

Metoden arbetar med kvadratiska residuer modulo en produkt av primtal.

Metoden har utmärkta statistiska egenskaper. Den genererar bitar (både till höger och vänster) som är fullständigt oförutsägbara för dem som inte känner p och q. Talet n kan ges ut publikt och metodens säkerhet bygger på att n inte kan faktoriseras.

Observera att det här gäller kvadratiska residuer modulo ett sammansatt tal. I fallet med primtal är frågeställningen "Är a en kvadratisk residu modulo p?" enkel.

Generatorn ger bara EN slumptalsbit per varv trots att beräkningar sker modulo n.

Effektiviteten kan förbättras genom att i varje varv använda $log\ log\ (n)$ bitar (beviset för detta är krångligt) utan att säkerheten minskas, så om $n \approx 10^{160}$ så erhålls 9 bitar i varje varv.

En $<k, m>$- PRBG är en avbildning $f : (Z_2)^k \rightarrow (Z_2)^m$, som kan beräknas inom polynomiell tid. Arguementet kallas frö ('seed') och värdet kallas en pseudoslumpbitsträng.

1.	Låt p och q vara två (k/2)-bitars primtal sådana att $p \equiv q \equiv 3 \pmod 4$ och sätt $n = p\ q$.
2.	Låt $s_0 \in$ QR(n) vara ett givet startvärde.
3.	Definiera $s_{i+1} = s_i^2\ mod\ n$.
4.	PRBG definieras av funktionen $f(s_0) = <z_1, ..., z_m>$, där $z_i = s_i\ mod\ 2$ för $1 \leq i \leq m$.
6.	Detta f är en (k, m)- PRBG.

Tabell 5.5. BBS PBRG

Exempel: Med n = 192 649 = 383 * 503 och $s_0 = 101355^2\ mod\ n$ = 20749 erhålls

i	s_i	z_i
0	20749	-
1	143135	1
2	177671	1
3	97048	0
4	89992	0
…	…	…

så slumptalsföljden börjar med 1 1 0 0 och fortsätter därefter med 1 1 1 0 0 0 0 1 0 0 1 ... .

Med hjälp av denna generator kan nu definieras följande probabilistiska chiffer.

5.5.2 Blums och Goldwassers PE

Grundidén är som följer. Initialvärdet s_0 genererar en sekvens slumpbitar $z_1, ..., z_m$. Ett chiffer framställs via $y = x$ **xor** z. Element $m + 1$, dvs s_{m+1}, bifogas chiffret.

Följande tabell beskriver metoden.

Parametrar	**Alfabet**
Talet $n = p\ q$, där p och q är primtal sådana att $p \equiv q \equiv 3 \pmod 4$. Talet n är publikt, p och q är privata.	Nyckelrymd $\mathtt{K} = <n, p, q>$. $K \in \mathtt{K}$ $x \in (Z_2)^m$ $r \in Z_n^*$ (slump-parameter) $\mathtt{M} = (Z_2)^m$ $\mathtt{C} = (Z_2)^m \times Z_n^*$
Chiffrering	**Dechiffrering**
Bestäm $z_1, ..., z_m$ från $s_0 = r$ enligt BBS $s_{m+1} = s_0^{l(m)} \bmod n$, där $l(m) = 2^{m+1}$ $y_i = (x_i + z_i) \bmod 2$, $i = 1, ..., m$. Definiera $y = <y_1, ..., y_m, s_{m+1}>$.	Bestäm $a_1 = ((p + 1) / 4)^m \bmod (p - 1)$ $a_2 = ((q + 1) / 4)^m \bmod (q - 1)$ $b_1 = s_{m+1}^{a_1} \bmod p$ $b_2 = s_{m+1}^{a_2} \bmod q$. Bestäm via CRT s_0 sådant att $s_0 \equiv b_1 \bmod p$ $s_0 \equiv b_2 \bmod q$ Bestäm $z_1, ..., z_m$ från s_0 via BBS. Bestäm klartexten via $x_i = (y_i + z_i) \bmod 2$, $i = 1, ..., m$.

Tabell 5.6. Blum-Goldwasser PE

Mottagaren erhåller chiffret genom att bestämma bestämma s_0 från s_{m+1}! Hur ?

Följande ger en bakgrund.

5.5.3 Kvadratiska residuer modulo ett sammansatt tal; igen

Om p och q är primtal och $n = p\ q$ så gäller för Jacobis symbol att

$J(x, n) = \pm 0$ om $gcd\ (x, n\ ) = 1$.

$J(x, n) = + 1$ om $L(x, p) = L(x, q) = 1$ eller om $L(x, p) = L(x, q) = - 1$.

$J(x, n) = - 1$ om en av $L(x, p)$ eller $L(x, q)$ är + 1 och den andra - 1.

Alltså är $x \in$ QR(n) precis då $J(x, p) = J(x, q) = 1$. Om faktoriseringen av n är känd är frågan om kvadratiska residuer lätt att besvara med givna primtal; annars inte.

Om $p \equiv q \equiv 3 \pmod 4$ så finns för varje kvadratisk residu x en entydig kvadratrot som också är en kvadratisk residu. En sådan kallas en *principalrot* till x.

Så, åter till frågan om att bestämma s_0 från s_{m+1}!

Varje s_{i-1} är en principalrot till s_i och $n = p\ q$ med $p \equiv q \equiv 3 \pmod 4$. Det betyder att kvadratrötterna till varje kvadratisk residu x modulo p lika med $\pm x^{(p+1)/4}$.

För Jacobis symbol gäller $J(x^{(p+1)/4}, p) = J(x, p)^{(p+1)/4} = 1$, varför $x^{(p+1)/4}$ är den principala roten.

Argumentera på samma vis, *mutatis mutandis*, för q.

Med 'Chinese remainder theorem' tas principalroten till x modulo n fram.

Lite generellare: $(x^{(p+1)/4})^{m+1}$ är den 2^{m+1}:te principalroten till x modulo p.

Eftersom Z_p^* har ordningen $p - 1$ så kan exponenten $((p + 1)/4)^{m+1}$ reduceras modulo $(p - 1)$ när $(x^{(p+1)/4})^{m+1}$ *mod* p beräknas.

Återigen analogt för q.

I tabell 5.6 visas hur dessa principalrötter m h a CRT kan användas för att finna den 2^{m+1}:te principalroten till s_{m+1} modulo n.

Exempel. Antag givet n = 192649 (som i BBS-exemplet), r = 20749 och klartexten

x = 11010011010011101101.

Nyckelsekvensen blir som ovan

z = 00011101010111010111,

varför chiffret blir

y = 00011101010111010111.

A beräknar också $s_{21} = s_{20}^2 \bmod n$ = 94739, som tillsammans med y sänds till B.

B vet faktoriseringen n = 383 * 503 och beräknar $(p + 1)/4 = 96$ och $(q + 1)/4 = 126$.

B beräknar vidare

$a_1 = ((p + 1)/4)^{21} \pmod{(p - 1)} = 266$,
a_2 = ...analogt med med q... = 486.

B beräknar därefter $b_1 = s_{21}{}^{a_1} \bmod p = 94739^{266} \bmod 383 = 67$ och b_2 = ... = 126.

B tar sedan fram s_0 genom att mha CRT lösa systemet

$s_0 \equiv 67 \pmod{383}$
$s_0 \equiv 126 \pmod{503}$

för att finna r = 20749. Med detta r kan B generera z och till sist dechiffrera: $x = y$ **xor** z.

Formellt kan följande definition sättas upp: tabell 5.7. Den kräver att begreppet ε-åtskiljbarhet är känt. Det är det dessvärre inte i denna kurs.

D
E
F
I
N
I
T
I
O
N

Ett probabilistiskt kryptosystem är en sextuple <M, C, K, E, D, R>, där

1. M är en klartextmängd
2. C är en chiffermängd
3. K är en nyckelmängd

4. För varje $k \in K$ är $e_k \in E$ en publik kryptering och $d_k \in D$ en privat dekryptering som uppfyller

   a. $e_k : M \times R \rightarrow C$ och $d_k: C \rightarrow M$ uppfyller

   $\forall\, x \in M$ och $r \in R$:: $d_k(e_k(x, r)) = b$

   b. För varje ε är sannolikhetsfördelningarna $p(c \mid K, x)$ och $p(c \mid K, x')$ icke ε-åtskiljbara, där $x, x' \in M$, $x \neq x'$ och $c \in C$.

Tabell 5.7. Probabilistiska chiffer

Noter. Skillnaden mellan ett probabilistiskt chiffer, t ex Blums och Goldwassers, och ett indeterministiskt, som t ex ElGamals, ligger bland annat i kravet på de betingade sannolikheterna $p(c \mid K, x)$ som gäller för de förra. Historiskt sett härstammar båda från mitten av 80- talet.

Ett chiffersystem kallas '*plaintext-aware*' om det är omöjligt att producera ett giltigt chiffer utan att känna klartexten. Sådana system kan implementeras genom att addera redundans till klartexten. Vid dechiffrering avslöjas om redundansen är korrekt.

Ett system kallas '*non-malleable*' (malleable = smidbar, foglig) om det givet $c = e(m)$ är omöjligt att producera ett giltigt chiffer c' svarande mot ett relaterat m'.

Ett chiffersystem kallas *semantiskt säkert* ('semantically secure') om, för alla sannolikhetsfördelningar över meddelanderymden, egenskaper hos klartexten givet chiffer också kan beräknas i polynomiell tid utan chiffret.

Egenskapen 'sematically secure' kan ses som en approximation av Shannons 'perfect secrecy' tillämplig på PKS, som ju aldrig kan uppfylla det senare.

5.6 Merkle-Hellman ränselchiffer ('knapsack cipher')

Öppna system kan framgångsrikt baseras på problem som är "svåra" ('NP-complete'). I denna klass av problem återfinns också det s k 'knapsack problem' eller 'subset sum problem'.

Merkle och Hellman föreslog 1978 hur man genom att definiera ett enkelt delproblem till det allmänna problemet kunde ge en kryptokonstruktör en 'trap door' för att ta fram sin privata nyckel.

Fastän denna metod alltså bygger på ett NP-fullständigt problem tog det inte mer än ett par år innan Shamir m fl lyckades kryptoanalysera denna metod och flera varianter av denna. Men: Medge att idéerna bakom metoden är eleganta!

Grundfrågeställningen är som följer.

> I uppsättningen $<s_1, ..., s_n, T>$ är komponenterna givna positiva heltal.
> Existerar det en Boolesk vektor $<x_1, ..., x_n>$ sådan att $\Sigma_i\, s_i\, x_i = T$?

Problemet kan alltså inte i det allmänna fallet lösas med en polynomiell algoritm. I följande specialfall är detta dock möjligt; nämligen om

Mängden $\{s_1, ..., s_n\}$ utgör en superökande lista: $s_j > \Sigma_{i \in [1, j-1]} s_i$.

Följande algoritm är uppenbarligen polynomiell, Ordo(n):

```
          snap (s, T)

   for i = n to 1 do
      if T ≥ si
      then T = T - si; xi = 1
      else xi = 0

   if T = 0 then "<x1, ..., xn> är lösningen"
   else "lösning saknas"
```

Figur 5.1. Lösning av ett 'superincreasing knapsack'

För **s** = $<s_1, ..., s_n>$ definieras tentativt $e_{\mathbf{s}}$: $\{0, 1\}^n \rightarrow \{0, 1, ... , \Sigma_i s_i\}$ via

$e_{\mathbf{s}}(\mathbf{x}) = \Sigma_i\ s_i\ x_i$.

Då erhålls visserligen en 1-1-avbildning. Denna är emellertid oduglig för kryptering eftersom en forcör kan använda algoritmen ovan.

Tanken är att omvandla **s** till en icke-superökande följd **t** med en privat transformation:

$t_i = a\ s_i\ mod\ p,$ (*)

där modulus $p > \Sigma s_i$ och $a \in [1, p - 1]$.

Listan **t** = $<t_1, ..., t_n>$ ges nu ut som publik nyckel medan a och p hålls privat.

Ett chiffer framställs nu alltså med (meddelandet uppfattas som binär vektor **x**)

$\mathbf{y} = e_{\mathbf{t}}(\mathbf{x}) = \Sigma_i\ t_i\ x_i$.

Motsvarande dechiffrering blir

$\mathbf{x} = d_{\mathbf{t}}(\mathbf{y}) = snap\ (\mathbf{s}, a^{-1}\mathbf{y}\ mod\ p)$ — Här tolkas bitvektorn **y** som ett heltal

Som sagt, elegant och bygger på NP-fullständighet, dock ändå forcerat (på en krypto-konferens) m h a linjär heltals- programmering och diofantiska ekvationer:

Av (*) framgår att att det finns heltal $k_1, ..., k_n$ sådana att

$$t_i u - k_i p = s_i, \qquad (**)$$

där $u = a^{-1} \bmod p$.

Då är $u / p - k_i / t_i = s_i / t_i p$.

Eftersom s_i är superökande, d v s $s_i < p2^{i-n}$, kommer, för små i, kvantiteterna k_i / t_i att ligga mycket nära varandra.

Från (**) följer (till exempel) att $| s_i k_1 - s_1 k_i | \leq p2^{i-n}$.

Några få (tre eller fyra) av dessa olikheter räcker för att bestämma $\{k_i\}$ och när dessa är kända är det lätt att forcera chiffret!

Noter

RSA-systemet beskrivs i [RSA78] och http://theory.lcs.mit.edu/~rivest/rsapaper.ps.

ElGamals chiffer är från [ElG85]. Elliptiska kurvor behandlades först i detta sammanhang i [Kob87]. Eb bra bok om kodningsteori är [Bla83]. Blum-Goldwasser-chiffret är från [BG85] och BBS-generatorn är från [BBS86].

Övningar

5.1. Betrakta ett synkront flödeschiffer med nyckelsekvens definierad av $k_i = (i + 1)^d \bmod n$, där d är den privata nyckeln i ett RSA-system och n är det publika modulovärdet. Chiffrering sker alltså med $c_i = m_i$ **xor** k_i.

Visa hur det i en känd klartext forcering är möjligt att bestämma k_3 och k_5 om de två paren $<m_1, c_1>$ och $<m_2, c_2>$ är kända. Om fler par är kända är det då möjligt att bestämma d och därmed göra en fullständig kryptoanalys?

5.2. Hur många fixpunkter ($c = m$) har systemet $c = m^e \bmod n$ då $n = p\,q = 5 * 7 = 35$ då

a. $e = 2$

b. $e = 3$

c. $e = 5$

d. $e = 7$

5.3. RSA bygger som bekant på atitmetik modulo n där n är en produkt av två olika primtal. Den berömda kryptoteknikern George F. Enigma har föreslagit en analog metod som bygger på produkten av tre olika primtal. Vad finns att säga om detta förslag ?

5.4. Låt $p = 23$ och $\alpha = 2$ vara parametrar för en ElGamal-kryptering. Antag vidare att $a = 9$ är den privata nyckeln associerad med den publika $\beta = \alpha^a = 6$. Kryptera meddelandet $x = 5$ med $k = 3$.

5.5. a. Hur många fixpunkter har det RSA-system som definieras av $n = p * q$ och $e = 3$? Motivera noggrant.

b. Analysera fallet då $p = 2p' + 1$, $q = 2q' + 1$, då p' och q' också är primtal.

5.6. Rabins PKS är som följer.

Modulen n är en produkt av två primtal p och q och $p \equiv q \equiv 3 \pmod 4$.

Alfabeten är $\mathtt{M} = \mathtt{C} = Z_n$ och $\mathtt{K} = \{<n, p, q, b> : 0 \le b \le n - 1\}$.

Värden n och b är publika medan p och q är privata.

Chiffreringen definieras av

$$e_K(x) = x\,(x + b) \bmod n.$$

a. Vad blir dechiffreringen ?

b. Tillämpa metoden med $n = 77$, $p = 7$, $q = 11$ och $b = 9$ genom att dechiffrera värdet 22.

5.7. Antag att $n = p\,q$, $0 < a < n$, x och y är kvadratrötter till a modulo n sådana att $y \neq x$ och $y \neq n - x$. Visa att $gcd(x + y, n) = p$ eller q.

5.8. Antag att B har ett RSA system med "godkända parametrar" p, q, n, e och d. Antag att A sänder ett meddelande till B i vilket varje alfabetiskt tecken (representerat i Z_{26}) krypteras separat.

a. Beskriv hur detta "protokoll" blir lätt att forcera!

b. Använd metoden för att forcera texten/chiffret

$$c = [365, 0, 4845, 14930, 2608, 2608, 0],$$

då $n = 18721$ och $e = 25$.

5.9. Visa att RSA-kryptering uppfyller den multiplikativa egenskapen $e_K(x)\,e_K(y) \equiv e_K(x\,y) \pmod n$. Vad säger denna ekvation i det fall då RSA används för att framställa digitala signaturer?

5.10. Visa att $561 = 3 * 11 * 17$ är ett Carlmichaeltal, d v s att

$$\forall a \in Z_{561}^{*}: a^{560} \equiv 1 \pmod{561}.$$

5.11. Låt $\mathbf{s} = <1, 3, 5, 10>$, $p = 20$ och $a = 7$ vara givna för ett Merkle-Hellman-chiffer.

a. Vad blir den "svåra" knap-sack vektorn **t** ?

b. Vad blir chiffret **y** svarande mot $\mathbf{x} = <1, 1, 0, 1>$?

c. Visa hur **y** dechiffreras till **x**.

5.12. ElGamals *chiffer* har implementerats i Z_{73} med primitivt element $\alpha = 5$ och publik nyckel $\beta = 49$.

Vilken är klartexten x som ger upphov till chiffret $<y_1, y_2> = <\alpha^k \bmod p, x\beta^k \bmod p> = <18, 7>$.

Ett "orakel" **A** som löser Diffie-Hellman-problemet, dvs beräknar $\alpha^{rs} = \mathbf{A}(p, \alpha, \alpha^r, \alpha^s)$, med givna argument finns tillgängligt och har givit följande resultat.

$\mathbf{A}(73, 5, 7, 18) = 9$
$\mathbf{A}(73, 5, 7, 49) = 27$
$\mathbf{A}(73, 5, 18, 49) = 8.$

5.13. De flesta metoder inom området PKS (chiffer, signaturer, nyckelhantering, identifiering, &c.) bygger på något eller några av följande problem.

a. Försök genomföra en analys som visar hur lösningen på dessa problem möjligen implicerar lösningar på andra av dessa problemen.

Problem	*Givet*	*Sökt*
FACTP (Faktorisering av heltal)	Ett heltal n.	Olika primtal p_i och heltal $e_i \geq 1$ så att $n = \Pi p_i^{ei}$.
RSAP (RSA-problemet)	Ett heltal $n = p\ q$ och $e > 0$ så att $gcd(e, \phi(n)) = 1$ och ett heltal c.	Ett heltal m så att $c = m^e \ mod\ n$.
SQRP (Kvadratrots-problemet)	Ett sammansatt heltal n och en kvadratisk rest a mod n.	Ett heltal x så att $x^2 \equiv a\ (mod\ n)$.
DLP (Diskreta logaritmen)	Ett primtal p och ett primitivt element α för Z_p^* och ett $\beta \in Z_p^*$.	Ett heltal x, $0 \leq x \leq p - 2$ så att $\alpha^x \equiv \beta\ (mod\ p)$.
DHP (Diffie-Hellman-problemet)	Ett primtal p och en generator α för Z_p^* och tal $\alpha^a \ mod\ p$ och $\alpha^b \ mod\ p$.	Heltalet $\alpha^{ab} \ mod\ p$.
SSP ('Subset-sum'-problemet)	Positiva heltal i mängden $S = \{a_1, ..., a_n\}$ och talet s.	Svaret på frågan: "Finns det en delmängd av S med summan s?"
QRP (Kvadratiska rester)	Ett udda sammansatt heltal n och ett heltal a med Jacobi-symbol $J(a, n) = 1$.	Svaret på frågan: "Är a en kvadratisk residu modulo n?"

Anm. Givet två problem A och B sägs A 'polytime reduce' till B,

$$A \leq_P B,$$

om det finns en algoritm som löser A och som använder en hypotetisk funktion som löser B och exekverar i polynomiell tid om algoritmen för B gör det.

Det betyder informellt att om A 'polytime reduce' till B så att B minst lika svårt som A.

Detta betyder, i sin tur, att om ett välstuderat problem A är svårt så ger ett bevis för att $A \leq_P B$ en stark indikation på att också B är svårt.

Om $A \leq_P B$ och $B \leq_P A$ så sägs A och B vara beräkningsmässigt ekvivalenta; $A \equiv_P B$.

Problemet handlar alltså om att fösöka definiera reltionerna $\leq_P$ och $\equiv_P$ över mängden

{FACTP, RSAP, SQRP, DLP, DHP, SSP, QRP}.

Not. Relationerna är inte fullständigt kända!

b. På vilket eller vilka problem bygger RSA, ElGamal och Blum-Goldwasser.

5.14. Betrakta en elleiptoisk kurva definierad av $y^2 = x^3 + x + 6$ över Z_{11}. Bestäm alla punkter på denna kurva.