Introduktion til Kryptologi

Mikkel Kamstrup Erlandsen

Indhold

1 Introduktion

Indledende skal vi have lidt notation på plads. $\mathbb{N}$ betegner mængden af *naturlige tal*; tælletallene $1, 2, 3, \ldots$ Vi skal ikke betragte 0 som et naturligt tal. De *hele tal* dvs. $\ldots, -2, -1, 0, 1, 2, 3, \ldots$ betegnes $\mathbb{Z}$. De *rationelle tal*, er mængden af brøker af hele tal, og betegnes $\mathbb{Q}$. På mængdeform har vi altså

$$\begin{aligned}\mathbb{N} &= \{1, 2, 3, 4, \ldots\} \\ \mathbb{Z} &= \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \\ \mathbb{Q} &= \{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\}\end{aligned}$$

1.1 Om Kryptologi

Kryptologi er et ord der dækker over flere ting. Der er to hovedemner indenfor moderne kryptologi, nemlig

Hemmeligholdelse Hemmelige beskeder så som banktransaktioner, militære ordrer og anden ømtålelig information, kan beskyttes ved at kode beskeden.

Autentitetsbekræftelse Sikring af at beskeden er modtaget som den er sendt. Altså at afgøre om beskeden er blevet ændret undervejs. Ligeledes at bekræfte, at senderen er den han udgiver sig for at være.

Man forbinder normalt kryptologi med hemmeligholdelse, men det er et faktum at autentitetsbekræftelse er et lige så vigtigt og stort felt. For at begrænse omfanget af denne note skal vi holde os til hemmeligholdelse. Autentitetsbekræftelse er forbundet med større matematisk- og konceptuel sværhed, og vil derfor være betragteligt uden for denne notes rækkevide.

1.2 Grundlæggende koncepter

Processen at tage en besked og kode, eller slører, den, så den er ulæselig for uvedkommende kaldes at *kryptere*. At afkode en krypteret besked kaldes at *dekryptere*. I kryptologi er der tre termer der er helt centrale, nemlig

Klartekst Den originale besked som ønskes sendt. Dette er altså typisk et hemmeligt brev eller de rå data til en banktransaktion.

Kryptotekst Den krypterede, dvs slørede, tekst. En ordentlig kryptotekst ser ud som en tilfældig blanding af bogstaver og/eller tal.

Nøgle Et tal eller en tekststreng der er brugt til at kryptere klarteksten med. Alt afhængigt af krypteringsmetoden kan det være den samme eller en anden nøgle der skal bruges til at dekryptere.

Eksempel 1.1: Vi starter helt elementært, for at få den grundlæggende tænkemåde klar. En person A ønsker at sende en besked til en person B. Beskeden kalder vi m (for det engelske *message*). A vil sende tekststrengen "`Hej med dig`". Dvs

$$m = \text{“}\texttt{Hej med dig}\text{”}$$

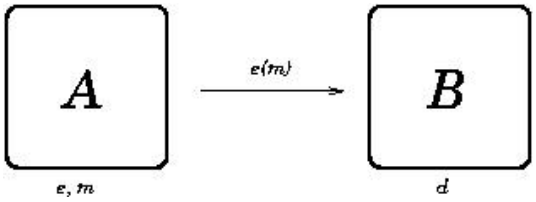
m er altså klarteksten. På sædvanelig grønspættemanér, udbytter A bogstaverne i m med det foregående i alfabetet (`a` krypteres til `å`). Denne operation kan opfattes som en funktion fra mængden af tekststrenge til sig selv, dvs noget der tager en tekststreng ind og spytter en ny ud. Kald denne funktion e (fra det engelske *encryption*), så har vi

$$e(m) = \text{“}\texttt{Gdi ldc chf}\text{”}$$

$e(m)$ er kryptoteksten, som A sender til B. Vi kan ligeldes lave en funktion der tager en tekststreng, og returnere en streng, hvor alle bogstaver er byttet ud med det næste i alfabetet (`å` dekrypteres til `a`). Kald denne funktion d (fra det engelske *decryption*) , vi har da

$$d(e(m)) = m$$

For at dette skal kunne fungere skal A altså kende m og e, og B skal kende funktionen d.



Figur 1.1: A kender beskeden m og krypterings-funktionen e. B kender kun dekrypetrings-funktionen d. A sender den krypterede besked $e(m)$ til B.

Følgende definition introducere flere vigtige begreber

Definition 1.2: Et *kryptosystem* består af en mængde af nøgler $\mathcal{K}$. Til enhver nøgle $k \in \mathcal{K}$ skal vi have en krypteringsfunktion e_k og en tilhørende dekrypteringsfunktion d_k, sådan at der for enhver klartekst m gælder at $d_k(e_k(m)) = m$.

Eksempel 1.3: I eksempel 1.1 kunne vi, istedet for at tælle én plads tilbage i alfabetet, tælle to eller flere pladser tilbage under krypteringen. Vi kunne

med andre ord bestemme os for et tal $k \in \mathbb{N}$ og så tælle k pladser tilbage i alfabetet for at kryptere et bogstav. Rammer man "a" mens man tæller baglæns, fortsætter man fra "å". Dekryptering foregår selvfølgelig ved at tælle k pladser frem igen, rammer man "å" fortsætter man fra "a".

Sætter vi mængden af nøgler til $\mathcal{K} = \mathbb{N}$ har vi et kryptosystem hvor krypteringsfunktionen e_k, for en nøgle $k \in \mathcal{K}$, er at tælle hvert bogstav i klarteksten k pladser tilbage. Dekrypteringsfunktionen d_k tæller k pladser frem. Læg mærke til at $k = 0$ (altså ingen kryptering) er en matematisk set lovlig kryptering. Da alfabetet har 29 bogstaver (husk "`w`") vil $k = 2$ og $k = 31$, give samme kryptering. Faktisk vil vi have samme kryptering for alle k på formen $k + 29p$, hvor $p \in \mathbb{Z}$. Effektivt set har vi altså kun 29 forskellige nøgler.

Dette kryptosystem er opkaldt efter den romerske kejser Caesar, og kaldes *Caesar-substitution*.

For $k = 4$ fås følgene tabel til beregning af e_4:

Klartekst	a	b	c	d	e	f	g	h	i	j	k	l	m	n
Kryptotekst	z	æ	ø	å	a	b	c	d	e	f	g	h	i	j

Klartekst	o	p	q	r	s	t	u	v	w	x	y	z	æ	ø	å
Kryptotekst	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Eksempel 1.4: Vi kan lave endnu et kryptosystem, der drastisk øger antallet af nøgler. Nemlig det såkaldte *permutations-kryptosystem* . Det beror på ideen at en tabel som ovenstående bestemmer en kryptering. Vi kunne udfylde kryptotekst-rækken med vores egen tilfældige blanding af bogstaver. En nøgle bliver således en tabel ala

Klartekst	a	b	c	d	e	f	g	h	i	j	k	l	m	n
Kryptotekst	x	t	k	d	v	y	æ	p	a	b	g	ø	c	j

Klartekst	o	p	q	r	s	t	u	v	w	x	y	z	æ	ø	å
Kryptotekst	i	o	å	z	l	w	n	h	e	f	m	q	s	u	r

Enhver konfiguration af nederste række giver en mulig nøgle. På den måde får vi $29! = 29 \cdot 28 \cdot 27 \cdots 2 \cdot 1$ forskellige nøgler. En hel del mere en Caesars 29. Krypterer vi f.eks. m fra eksempel 1.1 får vi med ovenstående nøgle

"`Hej med dig`" $\mapsto$ "`Pvb cvd daæ`"

1.3 Bogstaver som tal

For at komme dybere ind i kryptologien er det nødvendigt at have et solidt matematisk grundlag. Til dette har vi også brug for en matematisk måde at håndtere tekst på. Det er oplagt at nummerere bogstaverne fra 0 til 28. Dette er dog ikke helt tilstrækkeligt hvis man vil kryptere mere komplicerede

beskeder. De fleste længere tekster indeholder bla. tal, mellemrum og andre tegn så som punktum, komma, plus- og minustegn osv. Dette er dog let at omgås ved f.eks. at liste alle tegn (inkl. mellemrum) man kunne få brug for, og nummerere dem fra 0 og opefter. Hvis alle er enige om en sådan nummerering er der ingen problemer.

For senere refenrence er her en tegntabel, til oversættelse imellem bogstaver og tal. Når vi kun er interesserede bogstaver (som vi for simplicitetens skyld oftest er) vil vi referere til denne som den *simple tegntabel*

Den simple tegntabel

Tegn	a	b	c	d	e	f	g	h	i	j	k	l	m	n
Num. repr.	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Tegn	o	p	q	r	s	t	u	v	w	x	y	z	æ	ø	å
Num. repr.	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Er man interesseret i mere komplicerede tekster med flere tegn, er det blot at udvide den simple tegntabel. F.eks. benytter computeren ofte en tegntabel der hedder ASCII, som indeholder 256 standard tegn og bogstaver. Et andet ofte udnyttet tegnsæt er UTF-8, som er stort nok til at indeholde de fleste kendte skriftsprog (inkl. japansk, kinesisk, kyrillisk osv.).

En anden løsning kommer gratis i form af computerens binære representation af alt data. Computeren lagrer alt som binære tal, dvs. en stor samling 1'er og 0'er, også kaldet *bits*. Vil man krypterer en datastrøm er det altså blot at blive enige om hvor store klumper man krypterer af gangen. Man kunne f.eks. vælge at bruge sin krypteringsfunktion på hver 256-lange bitstreng[1] (fortolket som et tal).

Eksempel 1.5: Vi kan prøve at representere ordet `abe` numerisk. En måde at gøre det på er, at skrive de tilsvarende tal op på listeform

$$\texttt{abe} \mapsto (0, 1, 4)$$

I praksis er det ofte smart at representere en tekststreng som ét tal. Da $014 = 14$ ikke umiddelbart viser om der er tale om `abe`, `be` eller `o`, skal man altså tænke sig for at få en oversættelse der er entydig. Det første skridt kunne være at vedtage at alle bogstaver skulle oversættes til tocifrede tal. Så $\texttt{abe} = (00, 01, 04) = 000104$. For ikke at miste informationen indeholdt i nullerne, kunne man vedtage at man altid delte klarteksten op i klumper af en længde på tre bogstaver. Således vil man vide, at hvis man modtager `104` vil der være tale om `000104`.

[1] Dette giver hvad der svarer til 32kb blokke.

Metoderne fra eksempel 1.5 er ikke dem der bruges i praksis, men er blot med for at demeonstrere at det kan lade sig gøre. Pointen er at vi fremover uden tab af generalitet kan antage at klarteksten er et tal. Hvis m er en klartekst, giver det altså god mening at skrive $x = m + 17$.

1.4 Kodebrydning

Vil man lave sikre kryptosystemer er det vigtigt også at kende til de forskellige teknikker der er til at bryde kryptosystemer. Dvs. tiltvinge sig klarteksten udfra kryptoteksten, uden at være i besiddelse af nøglen. I fagsprog taler man om et *angreb* af kryptosystemet.

1.4.1 Angreb på Caesar-substitution

Det kommer næppe som nogen overraskelse at Caesar-substitution ikke er særligt sikker. Den har flere svagheder, men den største er nok det begrænsede antal nøgler. For at være præcis er der kun 29 forskellige nøgler. En computer vil have afprøvet alle muligheder på et splitsekund. Ved at sammenligne med en ordbog, vil den vide hvornår klarteksten var fundet.

1.4.2 Find "e"!

Skulle man bryde en Caesar-substitution i hånden, vil det heller ikke give de store kvaler. Der findes nemlig et trick, der kan bruges imod mange simple kryptosystemer. Tricket beror på det faktum at "e" er det hyppigst forekommende bogstav i de fleste sprog.

Vil man angribe en Caesar-krypteret tekst, starter man med at lede efter det hyppigst forekommende bogstav i kryptoteksten. For at være konkrete kan vi sige at det er "r". Man skal tælle 15 pladser baglæns, fra "e" for at ramme "r". Den hemmelige nøgle er altså med stor sandsynlighed 15. Hvis man får en meningsfuld tekst ved at dekrypterer med $k = 15$ er man så godt som sikker på at have fundet den rigtige nøgle.[2]

Tabel 1.1 viser fordelingen af bogstaver i det engelske sprog. Denne fordeling er ikke helt ulig dansk og mange andre vestlige sprog. Man kan lave en hyppighedsanalyse af alle bogstaverne i en kryptotekst og sammenligne med denne tabel. På denne måde kan man lave kvalificerede gæt på hvilke bogstaver i kryptoteksten der svarer til hvilke bogstaver i klarteksten. Dette vil være nok til at bryde permutations-kryptosystemet fra eksempel 1.4 uden at skulle afprøve 29! nøgler.

[2]Kan du finde en tekststreng der giver mening under dekryptering med to forskellige

Bogstav	Sands.	Bogstav	Sands.
A	.082	N	.067
B	.015	O	.075
C	.028	P	.019
D	.043	Q	.001
E	.127	R	.060
F	.022	S	.063
G	.020	T	.091
H	.061	U	.028
I	.070	V	.010
J	.002	W	.023
K	.008	X	.001
L	.040	Y	.020
M	.024	Z	.001

Tabel 1.1: Sandsynlighedsfordelingen for bogstaverne i det engelske alfabet. [*Stinson*, side 26].

Observation 1.6: Det ovenfor beskrevne angreb udfra analyse af bogstavforekomster i kryptoteksten beror på ét afgørende faktum. *Kryptosystemet under angreb krypterer altid enhver forekomst af et givet bogstav til det samme, uanset placering i teksten.* Altså er det vigtigt at designe sit kryptosystem så dette ikke sker!

1.5 Kerckhoffs princip

Det er en upræget overbevisning at man kan sikre sig ved at holde sin krypteringsmetode hemmelig. Historien har dog gang på gang vist at det *ikke* er tilfældet[3]. I kryptologien er der derfor et vigtigt princip, som man altid bør holde sig i mente, når man skal vurdere sikkerheden af et kryptosystem.

Kerckhoffs princip: En angriber, kender det anvendte kryptosystem.

Det omvendte; at man opnår sikkerhed ved at hemmeligholde hvilket kryptosystem man benytter, kaldes ofte for *security by obscurity*.

Det er en længere historie at komme ind på hvorfor security by obscurity ikke er sikker, og vi vil derfor ikke komme ind på det her.

nøgler?

[3]Det kan bla. nævnes at Tyskernes krypteringsmaskine, under anden verdenskrig, Enigma, byggede delvist på dette. Det fik afgørende betydning for D-dag.

til at finde krypteringsnøglen (a, b). Det oplyses at det hyppigste bogstav i klarteksten er `e` og det næsthyppigste er `d`.

```
(11,29 23,31,9,11,27,11,9,12,59,29,57,15,25,11,37,
       5,27,51,9,11,39, 25,11,41)
```

Hint: Der skal bruges to punkter for at finde ligningen for en ret linie.

(c) Hvad står der?

2 Talteori

Indtil videre har vi hovedsagligt holdt os til at motivere og styrke intuitionen. Matematiken har spillet en mindre rolle. For at få kryptologien ind i strengt matematiske rammer, er det nødvendigt med en del talteori. I dette afsnit præsenteres de fornødne resultater.

Da denne note handler om kryptologi, og ikke talteori, vil beviserne for sætningerne i dette afsnit blive underspillet, imens de praktiske anvendelser vil være af primær interesse. En mere grunddig introduktion til talteori kan findes i [*Hansen & Spalk*].

2.1 Division med rest

Teorien om endelige talsystemer er det redskab vi har brug for, for at behandle det komplicerede sammenspil imellem tal og bogstaver. Først skal vi vide lidt om moduloregning, dvs læren om division med rest. Vi starter med et vigtig resultat, som de fleste kender

Sætning 2.1: *Givet $n, p \in \mathbb{Z}$, $p \neq 0$, findes der, entydigt bestemte $q, r \in \mathbb{Z}$, hvor $0 \leq r < p$, så*

$$n = pq + r \tag{2.1}$$

Vi siger at p går q gange op i n med rest r.

Eksempel 2.2: Sætning 2.1 siger egentlig bare at det kan lade sig gøre at lave division med rest. Tager vi f.eks. $n = 9$ og $p = 4$, ved vi at 4 går to gange op i 9 med én til rest, dvs

$$9 = 4 \cdot 2 + 1$$

Med notation som i sætning 2.1 har vi altså $q = 2$ og $r = 1$. Det er klart at der ikke findes andre par der opfylder $9 = 4q + r$, hvor $0 \leq r < q$. Vi kunne f.eks. prøve med $9 = 4 \cdot 1 + 5$, men så er $r \not< q$.

Definition 2.3: For $n, p \in \mathbb{Z}$ og $p \neq 0$ skriver vi $p|n$, betydende "p går op i n". Dvs der findes et helt tal q, så $pq = n$. Altså at r er 0 i sætning 2.1.

Definition 2.4: For $n, m \in \mathbb{Z}$ og $p \in \mathbb{N}$ skriver vi

$$n \equiv m \mod p$$

hvis n og m har samme rest (givet fra sætn. 2.1) ved division med p. Vi siger at n er kongruent med m modulo p.

Eksempel 2.5: Et par simple eksempler skulle hjælpe til at forstå definition 2.4.

$$10 \equiv 13 \mod 3$$

Idet både 10 og 13 har rest 1 ved division med 3. Dvs. $10 \equiv 13 \equiv 1 \mod 3$. Overvej selv rigtigheden af følgende udsagn

$$3 \equiv 11 \mod 2 \qquad 7 \equiv 25 \mod 9$$
$$1 \not\equiv 0 \mod 2 \qquad 5 \not\equiv 9 \mod 3$$

Følgende sætning kan hjælpe os med at afgøre hvornår to tal er kongruente

Sætning 2.6: *Givet $p \in \mathbb{N}$ og $m, n \in \mathbb{Z}$ gælder at $m \equiv n \mod p \iff p|m-n$.*

Bevis $\Rightarrow$: Antag at $m \equiv n \mod p$. Da har m og n samme rest r, ved division med p. Der findes altså tal $q, z \in \mathbb{Z}$ så

$$m = pq + r$$
$$n = pz + r$$

Trækker vi den nederste ligning fra den øverste får vi

$$\begin{aligned} m - n &= pq - pz \\ &= p(q - z) \end{aligned}$$

Det betyder jo præcist at vi kan skrive $m - n$ som p gange et tal, altså at $p|m-n$.
$\Leftarrow$: Antag at $p|m-n$. Dvs. der findes et $q \in \mathbb{Z}$ så

$$\begin{aligned} pq &= m - n \qquad \Rightarrow \\ pq + n &= m \end{aligned}$$

Da pq har rest 0 ved division med p, må m og n have samme rest ved division med p. □

2.2 Restklasser

Vælger vi et fast $p \in \mathbb{N}$, kan vi, for ethvert $n \in \mathbb{Z}$, lave en (uendelig) liste af tal der er kongruente med $n \mod p$. Betegner vi mængden af alle tal der er

kongruent med $n \mod p$ med $[n]_p$, har vi for $p = 5$

$$\begin{aligned}
[0]_5 &= \{\ldots, -10, -5, 0, 5, 10, 15, \ldots, 5q + 0, \ldots\} \qquad q \in \mathbb{Z} \\
[1]_5 &= \{\ldots, -9, -4, 1, 6, 11, 16, \ldots, 5q + 1, \ldots\} \\
[2]_5 &= \{\ldots, -8, -3, 2, 7, 12, 17, \ldots, 5q + 2, \ldots\} \\
[3]_5 &= \{\ldots, -7, -2, 3, 8, 13, 18, \ldots, 5q + 3, \ldots\} \\
[4]_5 &= \{\ldots, -6, -1, 4, 9, 14, 19, \ldots, 5q + 4, \ldots\} \\
[5]_5 &= \{\ldots, -5, 0, 5, 10, 15, 20, \ldots, 5q + 5, \ldots\} = [0]_5 \\
[6]_5 &= \{\ldots, -4, 1, 6, 11, 16, 21, \ldots, 5q + 6, \ldots\} = [1]_5 \\
&\vdots
\end{aligned}$$

Dette motiverer os til følgende definition

Definition 2.7: For $n \in \mathbb{Z}$ og $p \in \mathbb{N}$ definerer vi *restklassen for n modulo p* til

$$[n]_p = \{\ m \in \mathbb{Z} \mid m \equiv n \mod p\ \}$$

Når det ikke giver anledning til forvirring vil vi blot skrive $[n]$.

Definition 2.8: Givet $p \in \mathbb{N}$ vil $\mathbb{Z}_p$ betegne mængden af restklasser modulo p.

Øverst i dette afsnit har vi set $\mathbb{Z}_5$, som består af fem restklasser.

$$\mathbb{Z}_5 = \{[0], [1], [2], [3], [4]\}$$

2.3 Modulær aritmetik

I dette afsnit vil vi hele tiden regne med kongruensklasser modulo p, og vil derfor, for at lette notationen, undertrykke p'et så vi blot skriver $[n]$.

Det smarte ved restklasserne indført i forrige afsnit er at man kan regne med dem, som var de tal. Vi skal give mening til hvad vi mener med udtrykkene

$$[n] + [m], \qquad \text{og} \qquad [n][m]$$

2.3.1 At regne med repræsentanter

Indtil nu er restklasser i $\mathbb{Z}_p$ blevet skrevet $[n]$. I den notation er der indbygget et underforstået tal $n \in \mathbb{Z}$, som man skal passe på. Ser vi abstrakt på et

element i $\mathbb{Z}_p$, har vi ikke noget på forhånd givet tal at skrive inden i [ ]. Vi ved blot at restklassen består af tal der alle er kongruente. Derfor vil det nogle gange blive nødvendigt at skrive en restklasse uden nogen reference til et bestemt tal. Hertil vil vi bruge store bogstaver, og f.eks. skrive $N \in \mathbb{Z}_p$.

Givet et $N \in \mathbb{Z}_p$, kan man vælge sig et element $a \in N$. Et sådan a kaldes en *repræsentant* for N. Hvis a og b er repræsentanter for restklasserne hhv. M og N defininerer vi

$$M + N = [a + b] \tag{2.2}$$
$$MN = [ab] \tag{2.3}$$

For at vi overhovedet kan bruge dette til noget, er det afgørende at operationerne er uafhængige af hvilken repræsentanter vi vælger. Tænk hvis vi valgte to forskellige repræsentanter a_1, a_2 for M, og b_1, b_2 for N, hvor $[a_1 + b_1] \neq [a_2 + b_2]$ – hvilken skulle så udnævnes til at være $M + N$? Vi har derfor følgende vigtige sætning

Sætning 2.9: *Hvis $M, N \in \mathbb{Z}_p$, er $M + N$ og MN uafhængige af valg af repræsentanter.*

Bevis Antag vi har to repræsentanter a_1, a_2 for M, og b_1, b_2 for N. Vi skal vise at $[a_1 + b_1] = [a_2 + b_2]$.Da $a_1 \equiv a_2$ og $b_1 \equiv b_2$, findes der, per sætning 2.6, q og z så

$$pq = a_1 - a_2$$
$$pz = b_1 - b_2$$

Lægger vi disse ligninger sammen fås

$$pq + pz = (a_1 - a_2) + (b_1 - b_2) \quad \Rightarrow$$
$$(a_1 + b_1) - (a_2 + b_2) = p(q + z)$$

Per sætning 2.6 må $(a_1 + b_1) \equiv (a_2 + b_2)$, og derfor $[a_1 + b_1] = [a_2 + b_2]$. At vise sætningen for multiplikation, er opgave 2.1 □

Eksempel 2.10: Når man regner med restklasser, har man altid konkrete klasser givet, og det er derfor ikke svært at vælge sine repræsentanter. Lader vi $p = 6$, og vil udregne $[2] + [1]$ kan vi blot vælge repræsentanterne 2 og 1. Så

$$[2] + [1] = [2 + 1] = [3]$$

Det overraskende kommer når man prøver med lidt andre tal. Udregner vi $[2]+[4]$ i $\mathbb{Z}_6$ (med repræsentanter 2 og 4) får vi

$$[2]+[4]=[2+4]=[6]=[0]$$

Her kommer det sidste lighedstegn fra sætning **??**, idet $6 \equiv 0 \mod 6$. $[2]+[4]$ giver altså $[0]$! Vi kunne have valgt andre repræsentanter end 2 og 4. F.eks. er 14 en repræsentant for $[2]$ og 22 en repræsentant for $[4]$ – check selv! Laver vi udregningen med disse repræsentanter får vi

$$[2]+[4]=[14+22]=[36]=[0]$$

– samme resultat. Dette er også hvad vi skulle forvente i lyset af sætning 2.9. Vi kan også prøve at gange nogle restklasser sammen.

$$[2][1]=[2\cdot 1]=[2] \qquad \text{og} \qquad [3][5]=[15]=[3]$$

Som vi har set med addition, kan to restklasser også ganges sammen så de giver nul

$$[2][3]=[6]=[0]$$

Der er flere gode gunde til at $\mathbb{Z}_p$ betegnes som det gør (udover hvad der er klart af definitionen). En vigtig ting er, at at der gælder regneregler i $\mathbb{Z}_p$, som vi kender dem i $\mathbb{Z}$. Dette ses af følgende sætning.

Sætning 2.11: *Lad $[a],[b],[c] \in \mathbb{Z}_p$. Da gælder følgende regneregler*

(i)

$$\begin{aligned}[a]+([b]+[c]) &= ([a]+[b])+[c]\\ [a]([b][c]) &= ([a][b])[c]\end{aligned}$$

(associative love)

(ii)

$$\begin{aligned}[0]+[a] &= [a]\\ [1][a] &= [a]\end{aligned}$$

(neutrale elementer)

(iii)

$$[a]([b]+[c])=[a][b]+[a][c]$$

(distributive lov)

(iv)

$$\begin{aligned}[a]+[b] &= [b]+[a]\\ [a][b] &= [b][a]\end{aligned}$$

(kommutative love)

Bevis Opgave 2.3 □

For at gøre analogien til $\mathbb{Z}$ færdig, har vi brug for at kunne trække to restklasser fra hinanden.

Definition 2.12: $[a] - [b] = [a] + [-b]$.

Observation: Denne definition af minus, sikrer at $[a] - [a] = [0]$.

Bemærkning 2.13: Vi har ingen *division* definineret. Hvis p er et primtal (se fodnoten s.17), kan man godt definere en division på $\mathbb{Z}_p$, men i almindelighed kan man ikke. Hele emnet om, hvornår man kan dividere i $\mathbb{Z}_p$, er spændende og relevant for vores emne, men af pladsgrunde må vi udelade det. Læg dog mærke til at man heller ikke kan dividerer alle tal i $\mathbb{Z}$. Vi har godt nok at $10/2 = 5 \in \mathbb{Z}$, men $11/2 \notin \mathbb{Z}$.

2.4 Eulers sætning

I dette afsnit vil vi formulere den sætning, der ligger til grund for meget af den mere avancerede del af kryptologien. Først er det dog nødvendigt at stifte bekendtskab, med en størrelse der dukker op mange steder i talteorien – nemlig Eulers ϕ-funktion.

2.4.1 Eulers ϕ-funktion

Definition 2.14: To tal $a, b \in \mathbb{N}$ siges at være *indbyrdes primiske*, hvis der ikke findes noget tal, andet end 1 der går op i både a og b.

Eksempel 2.15: Ser vi f.eks. på 6 og 9 er de altså *ikke* indbyrdes primiske da 3 går op i både 6 og 9. Til gengæld er 6 indbyrdes primisk med 7. Bemærk at et tal ikke er indbyrdes primisk med sig selv.

Definition 2.16: Givet $n \in \mathbb{N}$ definerer vi *Eulers ϕ-funktion*, $\phi(n)$ til at være antallet af tal $\leq n$, der er indbyrdes primiske med n.

Det er en god ide selv at checke rigtigheden af tabel 2.1

Bemærkning 2.17: Hvis $p \in \mathbb{N}$ er et primtal, er det let at beregne $\phi(p)$. Et primtal er jo netop defineret ved at der ikke er andre tal end 1 og tallet selv, der går op i det. Alle naturlige tal $< p$ er altså indbyrdes primisk med p. Dvs

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

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8

Tabel 2.1: $\phi(n)$ for $n \leq 15$.

Vi skal senere få brug for følgende sætning, der udvider bemærkning 2.17

Sætning 2.18: *Hvis p og q er to forskellige primtal gælder at*

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

BEVIS Udelades. Se evt. [*Hansen & Spalk*]. □

2.4.2 Eulers sætning

Sætning 2.19: *Lad a være indbyrdes primisk med n. Da gælder at*

$$[a]^{\phi(n)} = [1] \quad i \ \ \mathbb{Z}_n$$

Med andre ord

$$a^{\phi(n)} \equiv 1 \mod n$$

BEVIS Udelades. Se evt. [*Hansen & Spalk*] □

Eksempel 2.20: Lader vi $n = 119 = 7 \cdot 17$ og $a = 8$, er n og a indbyrdes primiske (det eneste der går op i 8 er 1, 2, 4 og 8). Nu siger Eulers sætning sammenkoblet med sætning 2.18 at

$$[8]^{\phi(7\cdot 17)} = [8]^{(7-1)(17-1)} = [8]^{96} = [1]$$

Hvis man havde gået i gang med at udregne $[8]^{96}$ på sin lommeregner, ville man have skulle udregne 8^{96}. Det er de færeste lommeregnere der kan gøre det.

Det fremgår dog endnu ikke klart hvorfor det er smart at kunne beregne sådanne størrelser. Det vil vi først se i afsnittet om RSA-kryptosystemet.

3 Symmetriske Kryptosystemer

Symmetriske kryptosystemer er den slags de fleste forbinder med kryptologi. De bygger på at der findes netop én nøgle der både krypterer og dekrypterer. To personer skal altså begge være i besiddelse af samme nøgle for at de kan kommunikere. Derfor kaldes sådanne systemer symmetriske.

Vi kan nu reformulere vores kryptosystemer fra afsnit 1, i et mere matematisk sprog. Indførelsen af $\mathbb{Z}_p$ i forrige afsnit er præcist det vi skal bruge for at sætte vores eksempler ind i en strengt matematisk kontekst.

Tallene i den simple tegntabel svarer til elementer i $\mathbb{Z}_{29}$, og vi vil derfor fremover blot betragte bogstaver som elementer i $\mathbb{Z}_{29}$.

Kryptosystemer

Vi har brug for en ny, og mere klar, definition af hvad et kryptosystem er. Fejlen med definition 1.2 er at den ikke er helt klar med hvad en *klartekst* er.

Definition 3.1: Et *(symmetrisk) kryptosystem* består af tre mængder

- $\mathcal{P}$ en mængde af mulige klartekster.
- $\mathcal{C}$ en mængde af mulige kryptotekster.
- $\mathcal{K}$ en mængde af mulige nøgler.

Til enhver nøgle $k \in \mathcal{K}$ skal vi have en krypteringsfunktion $e_k : \mathcal{P} \to \mathcal{C}$, og en dekrypteringsfunktion $d_k : \mathcal{C} \to \mathcal{P}$. For enhver klartekst $m \in \mathcal{P}$ skal der gælde at $d_k(e_k(m)) = m$.

Notation: Notationen $e_k : \mathcal{P} \to \mathcal{C}$, betyder at e_k er en funktion fra $\mathcal{P}$ til $\mathcal{C}$. Altså at e_k tager en klartekst $m \in \mathcal{P}$ ind og returnere en kryptotekst $e_k(m) = c \in \mathcal{C}$.

Bemærkning: Bogstaverne $\mathcal{P}$, $\mathcal{C}$ og $\mathcal{K}$ er ikke tilfældigt valgte. $\mathcal{P}$ kommer fra det engelske ord for klartekst, nemlig *plaintext*. Ligeledes kommer $\mathcal{C}$ af det engelske *ciphertext*, som betyder kryptotekst. $\mathcal{K}$ kommer af det engelske *keys*.

Vi vil nu se på hvordan vores kryptosystemer fra afsnit 1, ser ud i denne definition.

3.1 Caesar-substitution

Definition 3.2 (Caesar-kryptosystemet): $\mathcal{P} = \mathbb{Z}_{29}$, $\mathcal{C} = \mathbb{Z}_{29}$, $\mathcal{K} = \mathbb{Z}_{29}$. For $[k] \in \mathcal{K}$, $[m] \in \mathcal{P}$ og $[c] \in \mathcal{C}$ sætter vi

$$e_k([m]) = [m] - [k] \quad \text{og} \quad d_k([c]) = [c] + [k]$$

Husk at alle beregninger nu foregår i $\mathbb{Z}_{29}$. Da $[0] = [29]$, får vi præcist at vi tæller videre fra `å`, når vi rammer `a`, mens vi tæller baglæns.

3.2 Det affine kryptosystem

I opgave 1.3 definerede vi det affine kryptosystem. Det har også en reformulering, i vores nye termer. Vi har dog først brug for følgende sætning, der ikke bevises. Bevis findes i [*Hansen & Spalk*] og [*Stinson*].

Sætning 3.3: *Hvis $m, p \in \mathbb{Z}$ er indbyrdes primiske, findes et element $[m]^{-1} \in \mathbb{Z}_p$, der opfylder at $[m][m]^{-1} = [1]$.*

Definition 3.4 (Affine kryptosystem): $\mathcal{P} = \mathbb{Z}_{29}$ og $\mathcal{C} = \mathbb{Z}_{29}$. Nøgler er par af restklasser $([a], [b])$, hvor a er indbyrdes primisk med p. For $([a], [b]) \in \mathcal{K}$, $[m] \in \mathcal{P}$ og $[c] \in \mathcal{C}$ sætter vi

$$e_k([m]) = [a][m] + [b] \quad \text{og} \quad d_k([c]) = [a]^{-1}([c] - [b])$$

3.3 Andre symmetriske kryptosystemer

Nu skal man ikke tro at man kender alle symmetriske kryptosystemer, fordi man kender dem vi hidtil har nævnt. Der findes yderst effektive, og hidtil ubrudte, symmetriske kryptosystemer. Blandt de vigtigste er AES og Trippel-DES[5], hvoraf den sidste bla. bruges i dankort-automater. AES er nyere, og opfattes af mange som mere sikker end Trippel-DES.

AES og Trippel-DES er dog for komplicerede til at komme nærmere ind på her. En fuld gennemgang af begge findes i [*Stinson*]. Begge disse systemer er det der kaldes blok-kryptosystemer.

[5]Oprindeligt var det et system der blot hed DES. Det blev dog brudt, og man ændrede det lidt, så man krypterede tre gange med to forskellige nøgler. Herefter blev det kaldt Trippel-DES. Det er dog normalt ikke smart at bruge dette trick (det kan faktisk forværre sikkerheden i nogle systemer). DES er speciel i den henseende.

3.4 Blok-kryptering

Hidtil har vi krypteret et bogstav af gangen. Det sætter visse begrænsninger på opfindsomheden. Vælger vi f.eks. at kryptere to bogstaver af gangen er det muligt at bytte om på dem, lægge dem sammen (i $\mathbb{Z}_{29}$-forstand) og andet. Her kommer konceptet blok-kryptering på banen.

Definition 3.5: Et kryptosystem kaldes et *blok-kryptosystem*, hvis alle klartekst elementer $m \in \mathcal{P}$ er tupler af samme størrelse. Altså

$$m = (x_1, x_2, \ldots, x_n)$$

antallet af elementer i m kaldes *blokstørrelsen*. Ligeledes skal kryptoteksten også bestå af tupler af samme størrelse. Blokstørrelsen af klar- og kryptotekst behøves ikke at være den samme.

Eksempel 3.6: Her skal vi se på en af de simpleste blok-kryptosystemer, nemlig *Vigenere-kryptosystemet*. Vi beslutter os for en blokstørrelse på 3, for både klar- og kryptotekst, men den kunne principielt være et vilkårligt tal. En klartekst m er altså en tre-tuppel

$$m = (x_1, x_2, x_3) \qquad x_1, x_2, x_3 \in \mathbb{Z}_{29}$$

x'erne kunne have ligget i en vilkårlig mængde, men $\mathbb{Z}_{29}$ er vores standard-eksempel. En nøgle i Vigenere-systemet er ligeledes et tre-tuppel

$$k = (k_1, k_2, k_3) \qquad k_1, k_2, k_3 \in \mathbb{Z}_{29}$$

Vi kan nu definere $e_k(m)$ og $d_k(c)$, $c = (y_1, y_2, y_3)$ til

$$e_k(m) = (x_1 - k_1,\ x_2 - k_2,\ x_3 - k_3)$$
$$d_k(c) = (y_1 + k_1,\ y_2 + k_2,\ y_3 + k_3)$$

Havde vi valgt blokstørrelse 1, havde vi haft en almindelig Caesar-substitution. Vigenere-systemet udvider altså blot det vi kender. Ville man sende beskeden `kryptologi` til en ven ville man dele beskeden op som

$$\underbrace{\texttt{kry}}_{m_1}\ \underbrace{\texttt{pto}}_{m_2}\ \underbrace{\texttt{log}}_{m_3}\ \underbrace{\texttt{i}**}_{m_4}$$

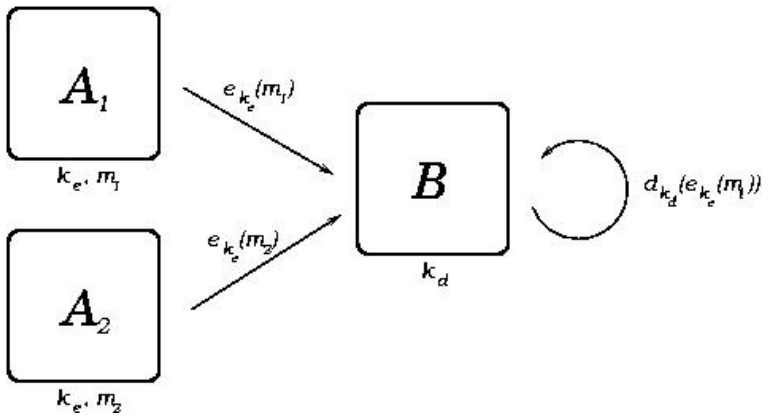
hvor $*$ udelades, og sende $e_k(m_1)$, $e_k(m_2)$,$e_k(m_3)$, i nævnte rækkefølge.

4 Asymmetriske Kryptosystemer

I et asymetrisk kryptosystem skal der ikke den samme nøgle til at kryptere og dekryptere (som det er tilfældet med symmetriske kryptosystemer). For at krypterer en besked bruger man en nøgle k_e, og til dekrypteringen k_d. Såfremt at det ikke er muligt at beregne k_d ud fra k_e, giver er det mulighed for at lave en helt ny måde at kommunikere på. Nemlig via den såkaldte *Public Key* - eller *Offentlig nøgle-kryptering.*

4.1 Offentlige nøgler

Ideen med offentlig nøgle-kryptering er at man har en offentlig nøgle som alle i princippet har adgang til, og en hemmelig, eller personlig, nøgle, som kun man selv kender. Alle der skal sende hemmelige beskeder til én, krypterer med den offentlige nøgle, og sender kryptoteksten. Når du modtager kryptoteksten dekrypterer du med den hemmelige nøgle, som kun du kender. Den offentlige nøgle kan altså *kun* bruges til at kryptere. Første gang man hører



Figur 4.1: To personer A_1 og A_2, kender B's offentlige nøgle k_e. De kan begge sende beskeder til B, men kun B selv kan dekrypterer dem med den hemmelige nøgle k_d.

om det, er det en underlig tanke, at man ikke skal bruge samme nøgle til at kryptere og dekryptere. Selve matematikken bag krypteringen bliver også mere vanskelig, men selve konceptet virker i praksis. Stort set alt kryptering på internettet foregår, ved hjælp af systemer som ligner dem vi vil gennemgå nu.

Bemærkning 4.1: Offentlig nøgle-kryptering virker faktisk *kun* under antagelse af Kerckhoffs princip.

4.2 RSA kryptosystemet

RSA står for Rivest, Shamir og Adleman som er hjernerne bag dette system, opfundet i 1977. Ideen om hele offentlig nøgle-konceptet stammer fra W. Diffie og M. Hellmann i 1976.

Konstruktion af nøgler

Da RSA er et asymmetrisk kryptosystem skal vi bruge et *nøgle-par*. Nemlig en offentlig og en hemmelig nøgle. Start med at vælge to store[6] primtal p og q. Dan nu

$$m = pq$$

Det er vigtigt at m har flere cifre end den maksimale længde af de tal vi krypterer. Typisk vælger man p og q så m har et ciffer mere end den typiske længde af klartekst-tallene. Vælg nu et k, indbyrdes primisk med $\phi(m) = (p-1)(q-1)$ (se sætning 2.18).

Offentlig nøgle: k, m

Hemmelig nøgle: $\phi(m)$

Kryptering

Både klartekst og kryptotekst skal være elementer i $\mathbb{Z}_m$. For en klartekst $[n] \in \mathbb{Z}_m$ definerer vi

$$e_k([n]) = [n]^k$$

Dekryptering

Dekryptering er lidt mere indviklet. Givet en kryptotekst $[n]^k \in \mathbb{Z}_m$, skal vi først beregne to hjælpestørrelser u og v, som løser ligningen

$$ku + (-\phi(m))v = 1 \tag{4.1}$$

Dette gøres i næste afsnit. Givet sådanne u og v har vi nemlig at

$$\begin{aligned}
([n]^k)^u &= [n]^{ku} \\
&= [n]^{1+\phi(m)v} \\
&= [n]([n]^{\phi(m)})^v \\
&= [n][1]^v \qquad \text{(Eulers sætning)} \\
&= [n]
\end{aligned}$$

Bemærk at vi for at bruge Eulers sætning bliver nød til at antage at m og n er indbyrdes primiske. Det kan vises at dette faktisk ikke er nødvendigt

[6] Med store menes mere end 250 cifre! I vores eksempler vil vi dog holde os til nogle mere håndterlige tal.

for at $[n]^{\phi(m)} = [1]$ når m er produktet af to primtal. Det kræver dog lidt talteori der ligge uden for vores nuværende rammer. Pointen er at det ikke er nødvendigt at kræve at klarteksten er indbyrdes primisk med m.

4.2.1 Beregning af u og v

Der er en generel metode til at finde hele tal u og v som løser ligningen

$$1 = au + bv$$

som i ligning (4.1). Metoden hedder *Euklids algoritme*. Det er dog et krav at a og b er indbyrdes primiske (se definition 2.14), for at løsninger findes.

Givet to hele tal r_0 og r_1, kan vi lave division med rest. Kald resten r_2

$$r_0 = r_1 q_1 + r_2$$

Dividerer vi r_1 med r_2, får vi en ny rest

$$r_1 = r_2 q_2 + r_3$$

Dette kan vi fortsætte med så vi får

$$\begin{aligned}
r_0 &= r_1 q_1 + r_2 \\
r_1 &= r_2 q_2 + r_3 \\
r_2 &= r_3 q_3 + r_4 \\
r_3 &= r_4 q_4 + r_5 \\
&\vdots \\
r_{n-2} &= r_{n-1} q_{n-1} + r_n \\
r_{n-1} &= r_n q_n + 0
\end{aligned}$$

I resten af dette afsnit vil r_n betegne den rest man får, umiddelbart inden man får en rest på nul. Dvs $r_{n+1} = 0$. Resten må nødvendigvis blive nul på et tidspunkt, da resterne bliver mindre og mindre. For at beregne u og v laver vi to hjælpefølger $u_0, u_1, u_2, \ldots, u_n$ og $v_0, v_1, v_2, \ldots, v_n$, hvor der gælder at $u_n = u$ og $v_n = v$. Sæt først

$$\begin{aligned}
u_0 &= 1 & \qquad v_0 &= 0 \\
u_1 &= 0 & \qquad v_1 &= 1
\end{aligned}$$

De næste led i følgen er defineret ved

$$\boxed{u_k = u_{k-2} - q_{k-1} u_{k-1} \qquad \text{og} \qquad v_k = v_{k-2} - q_{k-1} v_{k-1}} \tag{4.2}$$

Påstand 4.2: $r_k = r_0u_k + r_1v_k$ *for* $0 \leq k \leq n$

Bevis Bemærk først at

$$r_0 = r_0u_0 + r_1v_0$$
$$r_1 = r_0u_1 + r_1v_1$$

Hvis vi kan bevise udsagnet "*Påstanden er rigtig for* $k-1$ *og* $k-2 \Rightarrow$ *påstanden er rigtig for* k", er vi godt på vej. Thi vi har sikret os at den er sand for 0 og 1, og derfor også 2. Dermed er påstanden rigtig for 1 og 2, og derfor også 3. Dette kan fortsættes i det uendelige, og påstanden må være sand for alle $k \in \mathbb{N}$. Denne slags resonement kaldes for et *induktionsbevis*.

Antag at påstanden gælder for $k-1$ og $k-2$ dvs.

$$r_{k-1} = r_0u_{k-1} + r_1v_{k-1} \qquad r_{k-2} = r_0u_{k-2} + r_1v_{k-2}$$

I vores oprindelige udtryk for r_k kan vi nu indsætte dette

$$\begin{aligned} r_k &= r_{k-2} - q_{k-1}r_{k-1} \\ &= (r_0u_{k-2} + r_1v_{k-2}) - q_{k-1}(r_0u_{k-1} + r_1v_{k-1}) \\ &= r_0(u_{k-2} - q_{k-1}u_{k-1}) + r_1(v_{k-2} - q_{k-1}v_{k-1}) \\ &= r_0u_k + r_1v_k \end{aligned}$$

Hermed er udsagnet "Påstanden er rigtig for $k-1$ og $k-2 \Rightarrow$ påstanden er rigtig for k" bevist, og hermed er påstan 4.2 bevist. □

Følgende påstand er det sidste vi mangler for at kunne beregne u og v.

Påstand 4.3: *Hvis* r_0 *og* r_1 *er indbyrdes primiske så er* $r_n = 1$.

Bevis Udelades. Dette er en (meget) lille del af en større teori om "største fælles divisore". Se evt. [*Hansen & Spalk*]. □

Sammenholder vi påstand 4.2 med påstand 4.3, har vi altså følgende sætning

Sætning 4.4: *Givet hele tal* r_0 *og* r_1 *der er indbyrdes primiske, og at resten ved gentagen division* $r_{n+1} = 0$*, har vi at*

$$r_n = 1 = r_0u_n + r_1v_n$$

Eksempel 4.5: Lad $a = 61$ og $b = 27$, så er a og b indbyrdes primiske. Vi vil finde u og v så

$$1 = 61u + 27v$$

For at beregne u og v skal vi have fat i hjælpestørrelserne u_k og v_k. Den letteste måde at overskue beregningerne på er at lave et tabel. Vi har umiddelbart

k	q_k	r_k	u_k	v_k
0	-	61	1	0
1		27	0	1

Resten r_2 findes nu ved at dividerer 61 med 27. Antallet af gange det går op er $q_1 = 2$.

$$61 = 27 \cdot 2 + 7$$

Ved at bruge ligning (4.2), side 24, kan vi beregne u_2 og v_2, til hhv. $1-2\cdot 0 = 1$ og $0 - 2 \cdot 1 = -2$. Dvs. vi har

k	q_k	r_k	u_k	v_k
0	-	61	1	0
1	2	27	0	1
2		7	1	-2

$r_2 = 7$ går $q_2 = 3$ gange op i $r_1 = 27$, med en rest $r_3 = 6$. Desuden får vi hjælpestørrelserne til $u_3 = 0 - 3 \cdot 1 = -3$ og $v_3 = 1 - 3 \cdot (-2) = 7$. Skemaet ser altså således ud

k	q_k	r_k	u_k	v_k
0	-	61	1	0
1	2	27	0	1
2	3	7	1	-2
3		6	-3	7

Fortsætter vi med at udfylde skemaet, efter samme recept, ser vi at $r_5 = 0$

k	q_k	r_k	u_k	v_k
0	-	61	1	0
1	2	27	0	1
2	3	7	1	-2
3	1	6	-3	7
4	6	1	4	-9
5		0		

Sætning 4.4 giver nu at $1 = r_4 = r_0u_4 + r_1v_4$, altså

$$1 = 61 \cdot 4 + (-9) \cdot 27$$

4.3 Eksempler på RSA-kryptering

Frygt ikke hvis du er godt forvirret på nuværende tidspunkt. Koncepterne og metoderne introduceret hidtil, er ikke trivielle, og kræver at man har set dem anvendt i praksis før man forstår præcist hvordan de virker.

Eksempel 4.6: En person A beslutter sig for at gøre det muligt for andre at sende ham krypterede beskeder. Han har tre muligheder: Enten have en lang liste over hvilke personer der har hvilke nøgler. Ellers kan han have én nøgle til alle, og herved løbe an på at ingen af de involverede udlevere nøglen til uvedkommende. Sidst kan han vælge offentlig nøgle-kryptering, hvor alle kan gøre hvad de vil med hans offentlige nøgle. A vælger offentlig nøgle-kryptering. Da han har læst denne note, kender han til RSA-kryptosystemet, og vælger at benytte det[7].

Først skal der genereres et nøglepar. Hertil skal der bruges to store primtal. De skal have en størrelse så deres produkt har et ciffer mere end der er i klarteksten. A vælger en blokstørrelse på to, derfor vil klarteksten typisk have fire cifre. Der skal altså bruges to primtal hvor deres produkt har fem cifre. A vælger

$$p = 331 \qquad q = 149$$

Hermed fås

$$m = pq = 331 \cdot 149 = 49319$$
$$\phi(m) = (331 - 1)(149 - 1) = 48840$$

A skal nu finde et k indbyrdes primisk med $\phi(m)$. Da 7 er et primtal, og 7 ikke går op i 41580 er de indbyrdes primiske, og A vælger dette som sit k. Vi har altså

Offentlig nøgle	**Hemmelig nøgle**
$k = 7$	$\phi(m) = 48840$
$m = 49319$	

A lægger nu trygt sin offentlige nøgle på sin hjemmeside, så alle kan sende krypterede mails til ham. En anden person B, beslutter sig for at benytte den nye mulighed, og prøver at sende en lille testbesked. Beskeden er

`kryptologi`

A skriver også på sin hjemmeside at han benytter den simple tegntabel, og har en blokstørrelse på to. Dvs B skal bryde sin tekst op som følger

$$\underbrace{\texttt{kr}}_{1017}\ \underbrace{\texttt{yp}}_{2415}\ \underbrace{\texttt{to}}_{1914}\ \underbrace{\texttt{lo}}_{1114}\ \underbrace{\texttt{gi}}_{0608}$$

[7]Der findes flere offentlig nøgle kryptosystemer end RSA. Et af de mest udbredte alternativer er *elliptosk kurve-kryptering*, der garanterer samme sikkerhed som RSA.

B skal nu regne restklassen modulo $m = 49319$ ud, og opløfte den i $k = 7$. Følgende regnerier foregår altså i $\mathbb{Z}_{49319}$

$[1017]^7$	$[2415]^7$	$[1914]^7$	$[1114]^7$	$[0608]^7$
$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$
$[12623]$	$[25159]$	$[46563]$	$[18108]$	$[45733]$

B sender nu den nederste række til A.

Når A nu skal i gang med at dekrypterer, skal han bruge hjælpestørrelserne u og v, der opfylder

$$ku + (-\phi(m))v = 1 \tag{4.3}$$

Dem kan han beregne på forhånd og gemme sammen med $\phi(m)$. A udfylder følgende tabel, vha. Euklids algoritme

k	q_k	r_k	v_k	u_k
0	-	-48840	1	0
1	-6978	7	0	1
2	1	6	1	6978
3	6	1	-1	-6977
4		0		

Vi har altså

$$u = -6977 \qquad\qquad v = -1$$

Vi kan verificere

$$\begin{aligned} &7 \cdot (-6977) + (-48840) \cdot (-1) \\ =& -48839 + 48840 = 1 \end{aligned}$$

Dekryptering foregår nu ved at opløfte kryptoteksten i u'te. Det er et problem at u er negativ, da vi ikke ved noget om at opløfte i negative potenser i $\mathbb{Z}_m$. Vi vil altså gerne finde en løsning til ligning (4.3), hvor u er positiv. Dette kan findes ved følgende lille trick

$$\begin{aligned} 1 =& ku + (-\phi(m))v \\ =& ku + (-\phi(m))v + k\phi(m) - k\phi(m) \\ =& k\underbrace{(u + \phi(m))}_{u'} + (-\phi(m))\underbrace{(v + k)}_{v'} \end{aligned}$$

Vi får altså en ny løsning $u' = u+\phi(m)$ og $v' = v+k$. A's endelige hjælpestørrelser bliver altså

$$u = -6977 + 48840 = 41863 \qquad\qquad v = -1 + 7 = 6$$

A udregner nu

$$\underbrace{[12623]^{41863} \parallel [1017]}_{\texttt{kr}} \quad \underbrace{[25159]^{41863} \parallel [2415]}_{\texttt{yp}} \quad \underbrace{[46563]^{41863} \parallel [1914]}_{\texttt{to}} \quad \underbrace{[18108]^{41863} \parallel [1114]}_{\texttt{lo}} \quad \underbrace{[45733]^{41863} \parallel [0608]}_{\texttt{gi}}$$

4.4 Hvorfor virker RSA?

"Hvad er det der gør RSA sikker?" – kunne man med rette spørge. Sikkerheden i RSA bygger på at der, for tilstrækkeligt store tal, gælder:

- Det er svært (læs: umuligt) at udregne $\phi(m)$, når man ikke kender p og q.
- Givet m er det svært at finde p og q

Hvis det viser sig at en af disse ikke holder, vil sikkerheden på internettet lide et alvorligt knæk.

Udregning af $\phi(m)$

Hvis man har $\phi(m)$ kan man hurtigt udregne hjælpestørrelserne u og v, og derved bryde systemet. I vores eksempel fra før, er det ikke svært at udregne $\phi(m)$, selv om man kun kender m. Tallene er simpelt hen for små til at der er nogen sikkerhed i systemet. En computer ville hurtigt finde alle tal der er indbyrdes primisk med m, simpelthen ved at prøve alle tal $< m$.

Humlen er at der ikke er nogen kendt metode, andet end at prøve alle tal, til at beregne $\phi(m)$. Hvis m har over 400 cifre tager det mange år for alle verdens computere at udregne $\phi(m)$. Finder nogen en metode til, effektivt, at beregne $\phi(m)$ vil man enten blive rig, eller blive dræbt inden man kunne offentliggøre den.

At finde p og q

Kan man ud fra m finde p og q, er man godt på vej til at knække systemet, thi man kan udregne $\phi(m) = (p-1)(q-1)$. At finde primfaktorer som det hedder er et stort emne inden for talteorien. Der er flere metoder kendte til at finde primfaktorer, men alle bliver ekstremt tidskrævende ved store tal[8]. Den naive måde at finde p og q på – altså at prøve alle primtal mindre end m (faktisk er $< \sqrt{m}$ nok), vil hurtigt knække vores eksempel, men vil være tidsspilde at prøve på større tal.

[8] De moderne metoder til primfaktorisering er yderst komplicerede og bygger på teorien om elliptiske kurver.

Fandt nogen en hurtig metode til at udregne primfaktorer, ville de som med ϕ, blive rige eller dræbt inden resultatet blev kendt.

Litteratur

[*Stinson*] Douglas R. Stinson *Cryptography, Theory and Practice*, Chapman & Hall/CRC 2002.

[*Hansen & Spalk*] J.P Hansen og H.G. Spalk *Algebra og talteori*, Gyldendal Uddannelse 2002.

Indeks