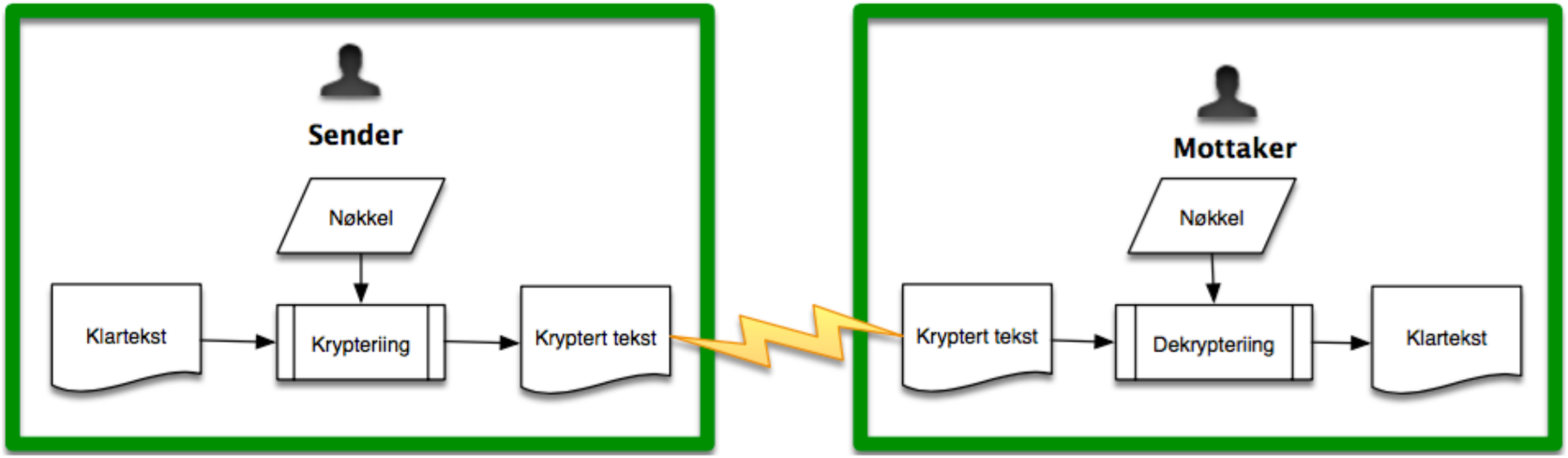
TDT4113

Oppgave 3: Krypto

Oversikt over oppgaven



- Implementere et sett av kryperingsalgoritmer som kontrolleres av en sender.
- Hver av disse tar en klartekst og en nøkkel, og “oversetter” teksten til en kryptert tekst (oversettelsen definert av nøkkelen)
- Mottaker har en algoritme og tilhørende nøkkel, som brukes for å “oversette” tilbake til klartekst.
- Dere skal også lage en hacker som brute-force’er for å knekke krypteringen.

Substitution - ciphers

- Den enkleste type ciphers er “substitusjons-cipher”. Her oversettes hvert symbol enkeltvis, for eksempel “A” blir “C”, “B” blir “F” etc.

- For å kryptere med slike algoritmer trenger vi en tabell som definerer byttene:

Klar-tekst:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Kodet tekst:	C	F	I	L	O	R	U	X	A	D	G	J	M	P	S	V	Y	B	E	H	K	N	Q	T	W	Z

- I stedet for å definere en tabell bruker vi gjerne en matematisk funksjon definert over heltallene *0, 1, 2, … N-1*, der *N* er antallet symboler i alfabetet. For eksempel er *N = 26* om vi begrenser oss til symbolene “A”, “B”, …, “Z”.

  - Erstatt symbol med verdi *i* med symbolet med verdi *foo(i)* for en gitt funksjon *foo()*. Symbolene verdi-sette null-indeksert, så “A”=0, “B”=1, …

Caesar-cipher

- For dette cipher'et er funksjonen for enkoding at bokstaven med verdi *i* blir oversatt til en verdi *i* + ***m*** mod *N*, der ***m*** er nøkkelen til krypteringen (merk at "mod *N*" er nødvendig for å sikre at alt er definert…).

Caesar-cipher med nøkkel 2																									
Klar-tekst:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Tall-verdi:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Kodet verdi:	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1
Kodet tekst:	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

- **Spørsmål:** Kan dere dekode meldingen "UWEEGUU" om dere vet at den er kodet med Caesar-cipher'et og nøkkel ***m*** = *2*?

Dekryptering av Caesar-cipheret

- Teksten "UWEEGUU" kan rimelig enkelt dekrypteres "SUCCESS" til når vi vet at den er kryptert med ***m*** = 2. Gå "baklengs" i tabellen for denne nøkkelen.

- En mer effektiv måte å gjøre det på er å ***kryptere*** teksten "UWEEGUU" med nøkkel ***n*** = 24. Nå blir den dobbelt-krypterte teksten "SUCCESS". Dette virker for Caesar så lenge ***n*** + ***m*** mod *N = 0;* her: *24 + 2* mod *26 = 26* mod *26 = 0*.

Caesar cipher med nøkkel 24																									
Klar-tekst:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Tall-verdi:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Kodet verdi:	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Kodet tekst:	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X

- Dekryptering er (implementasjonsteknisk) trivielt så lenge vi matcher nøklene ***n*** og ***m***.

Multiplikasjonscipher

- Caesar: Bytt bokstav med "verdi" *i* til en ny bokstav med verdi *i* + ***m*** mod *N*.
- Multiplikasjonscipheret bruker multiplikasjon, ikke addisjon: *i* —> *i* • ***m*** mod *N*.

Multiplikasjons-cipher med nøkkel 3																									
Klar-tekst:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Tall-verdi:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Kodet verdi:	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23
Kodet tekst:	A	D	G	J	M	P	S	V	Y	B	E	H	K	N	Q	T	W	Z	C	F	I	L	O	R	U	X

Multiplikasjons-cipher med nøkkel 2																										
Klar-tekst:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Tall-verdi:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Kodet verdi:	0	2	4	6	8	10	12	14	16	18	20	22	24	0	2	4	6	8	10	12	14	16	18	20	22	24
Kodet tekst:	A	C	E	G	I	K	M	O	Q	S	U	W	Y	A	C	E	G	I	K	M	O	Q	S	U	W	Y

- Dermed: Ikke alle nøkler ***m*** kan benyttes, må finnes en ***n*** slik at ***m*** • ***n*** mod *N* = *1*. Funksjonen `modular_inverse` i hjelpe-koden kan være nyttig her.

Affine

- Dette er en kombo av multiplikasjon og addisjon: $i \longrightarrow i \bullet \boldsymbol{m}_1 + \boldsymbol{m}_2$
  - Først multiplikasjon
  - Deretter Caesar
- Pass på å gjøre operasjonene i motsatt rekkefølge ved de-krypteringen…
  - Først Caesar med nøkkel $\boldsymbol{n}_2$ - som er valgt så den matcher $\boldsymbol{m}_2$.
  - … deretter multiplikasjon med nøkkel $\boldsymbol{n}_1$ som matcher $\boldsymbol{m}_1$.
- Her er altså nøkkelen $\boldsymbol{m}$ en tuple bestående av ($\boldsymbol{m}_1$, $\boldsymbol{m}_2$), og tilsvarende for $\boldsymbol{n}$.

“Unbreakable”

- Å bryte et cipher som alltid bytter en “A” med et fast symbol, “B” mer et annet fast symbol, etc., er ganske enkelt vha frekvens-tabeller
- På 1700-tallet fant man derfor på et cipher som ikke har en fast substitusjons-tabell; denne ble ansett å være ubrytelig
- Nøkkel er et ***ord***
- Kryptering av “HEMMELIGHET” med kode-ord “PIZZA”:

“Ubrytelig” cipher med nøkkelord PIZZA											
Klar-tekst:	H	E	M	M	E	L	I	G	H	E	T
Tall-verdi:	7	4	12	12	4	11	8	6	7	4	19
Nøkkelord:	P	I	Z	Z	A	P	I	Z	Z	A	P
Nøkkel-verdi:	15	8	25	25	0	15	8	25	25	0	15
Kodet verdi:	22	12	11	11	4	0	16	5	6	4	8
Kodet tekst:	W	M	L	L	E	A	Q	F	G	E	I

Letter	Relative frequency in the English language
a	8.167%
b	1.492%
c	2.782%
d	4.253%
e	12.702%
f	2.228%
g	2.015%
h	6.094%
i	6.966%
j	0.153%
k	0.772%
l	4.025%
m	2.406%
n	6.749%
o	7.507%
p	1.929%
q	0.095%
r	5.987%
s	6.327%
t	9.056%
u	2.758%
v	0.978%
w	2.360%
x	0.150%
y	1.974%
z	0.074%

Letter	Relative frequency as the first letter of an English word
a	11.602%
b	4.702%
c	3.511%
d	2.670%
e	2.007%
f	3.779%
g	1.950%
h	7.232%
i	6.286%
j	0.597%
k	0.590%
l	2.705%
m	4.383%
n	2.365%
o	6.264%
p	2.545%
q	0.173%
r	1.653%
s	7.755%
t	16.671%
u	1.487%
v	0.649%
w	6.753%
x	0.017%
y	1.620%
z	0.034%

RSA

- Ulempen med cipherne så langt er:
  - … at de er enkle å knekke
  - … og at sender og mottaker må være enige om matchende nøkler
- RSA unngår dette. Mottaker oppgir krypterings-nøkkel han ønsker brukt; det å generere dekrypterings-nøkkel fra denne er komputasjonelt vanskelig.

RSA: Nøkler og basis-algoritme

- Nøklene defineres slik:
  - Generer to (tilfeldige) primtall, ***p*** og ***q***. Se hjelpe-koden, funksjon `generate_random_prime`.
  - Definer ***n*** = $p \bullet q$, ***ϕ***=$(p-1) \bullet (q-1)$ og et tilfeldig tall ***e***, $2 < e < \phi$.
  - Til slutt definerer vi ***d*** slik at $d \bullet e \bmod \phi = 1$, se i `modular_inverse` i hjelpe-koden.
  - Nå er *(n, e)* den **offentlige** nøkkelen for å kryptere, *(n, d)* den **hemmelige** nøkkelen mottaker bruker for å de-kryptere.
- Vi håndterer en melding bestående av ett (unsigned) heltall *t* slik:
  - Kryptering: $\boldsymbol{c} = t^e \bmod n$; i Python skriver vi `c = pow(t, e, n)`.
  - Dekryptering: $\boldsymbol{t'} = c^d \bmod n$; her vil det så holde at $t' = t$.

RSA - Fra symboler til hel-tall

- (Vår versjon av) RSA-algoritmen tar ***positive heltall*** og krypterer disse.
- Input er en ***tekst-streng***. —> Vi trenger en måte å "oversette" fra tekst til heltall
- **Algoritme:**
    1. Del tekst-strengen opp i blokker
    2. For hver blokk:
        1. For hvert symbol i en blokk: Finn ASCII verdi i binær-format
        2. Slå sammen alle "binær-strengene" og finn desimaltall-verdien
- En tekst blir dermed "oversatt" til en liste heltall
- Se funksjonen `blocks_from_text` i hjelpe-koden for denne funksjonaliteten.
- Det er en tilsvarende `text_from_blocks` som kan brukes ved de-kryptering.

	Blokk 1			Blokk 2		
Tekst:	P	Y	T	H	O	N
ASCII:	80	89	84	72	79	78
Binær:	010100000101100101010100			01001000 0100111101001110		
Tall-verdi:	**5265748**			**4738894**		

Eksempel:

- Input-tekst "Python", blokk-størrelse 3
- For Blokk 1:
    - "PYT" gir ASCII-verdiene 80, 89, 84
    - I binær-enkoding, og satt sammen, får vi `010100000101100101010100`
    - Dette er `5265748` i desimal.
- For Blokk 2:
    - Tilsvarende skritt gir tall-representasjonen `4738894` for blokken "HON".
- Totalt vil "PYTHON" dermed bli listen `[5265748, 4738894]`.

Hacker

- Denne karen får fatt på kryptert melding, og forsøker å brute-force de-krypteringen når vi antar at krypterings-*algoritmen* er kjent.
- Han vurderer en de-krypteringsnøkkel ved å se på ordene i sin “plain-text”, og sammenligne med ordene i en ordliste (se filen `english_words.txt` som ligger på its)
- Hvis han får en full match er han ferdig, hvis ikke fortsetter han og viser til slutt beste resultat.
- Hackeren skal kunne knekke ALLE cipher’ene bortsett fra RSA.
- **Eksempel:**
  - Har fanget cipher-meldingen “UWEEGUU” kodet med Caesar.
  - Prøver med dekrypterings-nøkler 1, 2, 3, … 25:
    - Nøkkel 1 gir “VXFFHVV”, som ikke finnes i ordlisten og vi fortsetter …
    - Nøkkel 2 gir “WYGGIWW”, som ikke finnes i ordlisten og vi fortsetter …
    - …
    - Nøkkel 24 gir “SUCCESS” som *finnes* i ordlisten. Ettersom klartekst bare er ett ord er vi ferdige!

Ting å tenke på

- Cipher:
  - Alle må kunne generere nøkkel-par, for eksempel *n* og *m* slik at *n + m mod N = 0* for Caesar-cipher
  - Ethvert cipher må kunne gjøre “sin” operasjon med en gitt nøkkel. Merk at det er samme operasjon (men forskjellige nøkler) for kryptering og de-kryptering.
  - Ethvert cipher må kunne fortelle “hacker’en” settet av mulige nøkler. For “Unbreakable” er dette alle ordene i `english_words.txt`, mens det for Caesar er alle heltall 1, 2, …, N-1 etc.
  - Alle cipher’ene har samme type funksjonalitet. —> Lag dem som sub-klasser av en generisk klasse `Cipher`, slik at et annet objekt som interagerer med et cipher ikke trenger vite *hvilket* det er. Denne abstraksjonen gjør programmet ditt “renere”…

Flere ting å tenke på…

- “Personer”:
  - Det er tre aktører involvert: Sender, Mottaker, og Hacker.
  - Alle tre jobber mot et cipher.
  - Alle tre har en tekst de vil “oversette” (fra klartekst til ciphertekst eller motsatt).
  - Alle tre har en nøkkel, evt et sett av mulige nøkler som de vil bruke når de jobber mot cipheret
  - Siden det er delt funksjonalitet mellom personene — uten at oppførselen deres er identisk — er det igjen nyttig å tenke sub-klassing!

Tilbakemelding via referanse-grp

Vi tar veldig gjerne tilbakemeldinger. Bruk 5 min på dette sammen med referansegruppa nå, så får vi viderebrakt info fra dem.

Ett nytt oppslag er lagt ut på **TDT4113 DATATEKN PROGR PROSJ HØST 2016**:

Dette er et rimelig nytt kurs, så vi forelesere er veldig interessert i tilbakemeldinger fra dere, være seg kommentarer, ris & ros, tips, eller ideer til morsomme oppgaver. Hvis dere vil kontakte oss direkte er e-post adressen tdt4113@idi.ntnu.no. Hvis dere heller ønsker å gjøre det anonymt, så send beskjed via referanse-gruppas medlemmer (se eget innlegg med e-post for dem)

.... TDT4113-foreleserne

Best Regards

its learning