Kryptering 1 Hösten 2023

Laboration

Instruktioner

Denna laboration är en av examinationsmomenten i kursen *Kryptering 1* och som sådan omfattar den 1 högskolepoäng. I arbetstid ska 1 högskolepoäng motsvara 25–30 timmar. Betygskalan är Godkänd/Underkänd. För betyget Godkänd krävs att

- laborationsuppgifterna på följande sidor redovisas enskilt (samarbete är tillåtet men som vanligt är plagiering **inte** tillåtet)
- man använder CoCalc eller SageMath (för mer information se bl a dokumenten *Kom igång med SageMath/CoCalc* och *Kryptering i SageMath*)
- varje uppgift är korrekt löst (se nästa avsnitt för information om vad som ska redovisas i respektive uppgift)
- lösningarna presenteras tydligt och strukturerat
- både källfil i form av antingen Sage worksheet (`sagews`) eller Jupyter (`ipynb`) samt en typsatt version (`pdf`) har lämnats in. Blanka mallfiler finns på Canvas.

Sista inlämningsdatum är **7 januari 2024**. Vid omexamination ersätts uppgifterna, med undantag för dem som eventuellt fick UX och ska komplettera sin inlämning. Sista inlämningsdatum för omexamination är **17 mars 2024** och **31 maj 2024**.

Vad som ska redovisas i respektive uppgift

På följande sidor hittar du fyra uppgifter. Du ska lösa tre av dem. Första och andra uppgiften är obligatoriska medan du får välja mellan att lösa tredje eller fjärde uppgiften. Din lösning ska i respektive uppgift innehålla följande.

1. Kod och svar i båda deluppgifterna. Koden behöver inte kommenteras om du använder funktionerna som finna i filen `kryptering1.sage`.
2. Svara med nyckel och de två första meningarna i klartexten. Beskriv hur du gick tillväga med förklarande text till koden. Moment som upprepas räcker att beskriva två-tre gånger.

   Tips: Spara en detaljerad beskrivning för egen del – när det gått en tid har du kanske glömt bort hur du gjorde.
3. Kod och svar ska redovisas.
4. Kod och svar ska redovisas.

I slutet av denna fil finns en bilaga med några ledningar till uppgifterna.

XSEUA KQXX!

Laborationsuppgifter

Uppgift 1 (Några klassiska kryptosystem och en lite mer modernare)

I denna uppgift anvnds följande krypteringsnycklar för respektive kryptosystem.

- Förskjutningskrypto med steget $k = 17$.
- Monoalfabetiskt substitutionskrypto med följande bijektion.

a	b	c	d	e	f	g	h	i	j	k	l	m	n
D	Q	K	Å	A	Ä	U	R	Ö	S	H	J	I	E
o	p	q	r	s	t	u	v	x	y	z	å	ä	ö
B	C	P	X	V	F	O	G	Y	L	M	Z	T	N

- Affint krypto med paret $(a, b) = (13, 9)$.
- Vigenèrekrypto med nyckelordet $n =$ `gömd`.
- Transpositionskrypto med permutationen $k = (2, 0, 3, 1)$.
- Playfair med följande polybioskvadrat.

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

- ADFGVX med matrisen och permutationen

	A	D	F	G	V	X
A	G	6	U	7	A	Q
D	8	R	D	S	H	J
F	I	X	1	F	2	P
G	L	W	5	M	T	O
V	E	B	4	V	9	O
X	C	Y	K	Z	N	3

respektive $p = (1, 4, 3, 0, 2)$.

- Enigma med kopplingstavlan

  A↔L, D↔Q, K↔N, M↔S, U↔X,

  samt rotor II, IV, I i nämnd ordning och satta i startläge SYD.
- Hillkrypto med matrisen

$$A = \begin{pmatrix} 5 & 2 & 18 \\ 19 & 1 & 9 \\ 17 & 16 & 13 \end{pmatrix}$$

  och krypteringsfunktionen $y \equiv Ax \pmod{28}$.
- Data Encryption Standard (DES) med huvudnyckeln

$$K = 1111000011110000111100001111000011110000111100001111000011110000.$$

  Tips: `K = 8 * (4 * [1] + 4 * [0])`

- RSA med paret

$$(n, e) = (36\,134\,934\,063\,919\,959\,150\,141\,797\,353\,966\,441, 1\,330\,643\,366\,620\,853\,071).$$

I filen `kryptering1.sage` finns funktioner definierade med vilka man kan kryptera och dekryptera meddelande med ovanstående nämnda kryptosystem.

(a) Kryptera klartexten `imponerande` med samtliga krypton ovan.

(b) Nedanstående kryptogram har erhållits med något av de aktuella kryptona.

I: `LIGSLAAÄTLMMTNUOIRLREDGM`

II: `LDEOBMRNJKMMLXYBRJQFPHQZPYIRD`

III: 32 107 833 669 138 743 416 991 214 827 014 308

IV: `RBBJDQDEÅBBJDQDEÅGAIHDEIDEJÖFDCZ`

V: `JÖBSTFKÄJJSAXLCEÖLUHÖACÄNMLÅMCAOIBTT`

VI: `FMJÖJTIBYKFGÖKLHÖFGQFÅMBDJÅFUUFMJKUFK`

VII: `1DCDB9E4E6EA1049 83B2AE7D8E20B92B 63C8B4251AFAC981 A6F5AED397A81135`

VIII: `IPGCÅCYVCPGÖRHIRU`

IX: `VDAFVFDDXVXVGXGAVDFDGAVX`

X: `RTBLMÖPLVKÄPGSQULHGDYEBHJ`

XI: `RVTWPUBAXNYCCPLWHR`

Para ihop kryptogram med rätt krypto och dekryptera kryptogrammen.

Uppgift 2 (Endast kryptogram känt-attack)

I filen `kryptogram.sage` finns det 31 stycken kryptogram, vilka var och en är krypterad med ett monoalfabetiskt substitutionskrypto. Vid kryptering av respektive kryptogram har olika nycklar använts. Även olika klartexten döljer sig bakom kryptogrammen.

Låt d vara den dag i månaden som du fyller år. Du ska knäcka nyckeln och bestämma klartexten genom att utföra en endast kryptogram känt-attack på det kryptogram som är sparat i variabeln `kryptogram_`d. Filen `kryptering1.sage` innehåller funktioner som får användas för att lösa uppgiften.

Uppgift 3 (Implementation av ett kryptosystem)

Beskrivning av kryptosystemet Vigenère-Affint-Hill-Feistel (VAHF)

Bokstäverna i en klartext kodas med funktionen sve: $\mathcal{A}_{\text{sve}} \to \mathbb{Z}_{28}$ och delas in i block av längd 8. Övriga bokstäver, siffror och tecken ignoreras vid kodning från textsträng till lista av heltal. Låt $\boldsymbol{m} = (m_1, m_2, \ldots, m_8)$, där $m_k \in \mathbb{Z}_{28}$, betecknar ett sådant block. Sätt

$$L_0 = (m_1, m_2, m_3, m_4) \quad \text{och} \quad R_0 = (m_5, m_6, m_7, m_8).$$

Bestäm sedan L_i, och R_i, där $i = 1, 2, 3$ enligt

$$L_i = R_{i-1} \quad \text{och} \quad R_i \equiv L_{i-1} + f(R_{k-1}, K_i) \pmod{28},$$

se figur 1. Då ges kryptogrammet av $\boldsymbol{c} = (c_1, c_2, \ldots, c_8)$ där

$$L_3 = (c_1, c_2, c_3, c_4) \quad \text{och} \quad R_3 = (c_5, c_6, c_7, c_8).$$

Det återstår att beskriva funktionen f och hur man genererar rundnycklarna K_i. Låt

$$P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{och} \quad P_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

samt

$$A = \begin{pmatrix} 11 & 7 & 19 & 0 & 1 & 15 \\ 2 & 9 & 27 & 5 & 21 & 5 \\ 25 & 8 & 12 & 2 & 4 & 10 \\ 20 & 16 & 4 & 18 & 1 & 23 \\ 3 & 0 & 19 & 21 & 2 & 13 \\ 6 & 24 & 22 & 5 & 24 & 26 \end{pmatrix}.$$

Då definieras funktionen f enligt

Steg 1: Sätt $E = P_1 R_{i-1}$, vilket motsvarar en permutation och expansion av R_{i-1} från fyra element till sex element.

Steg 2: Sätt $X \equiv E + K_i \pmod{28}$, d v s addera rundnyckeln K_i till föregående resultat.

Steg 3: Sätt $Y \equiv AX \pmod{28}$, d v s kryptera X med ett Hillkyrpto.

Steg 4: Sätt $Z = P_2 X$, vilket motsvarar en permutation och kontraktion av Y från sex element till fyra element.

Steg 5: Returnera Z.

Med andra ord kan f sammanfattas på formen

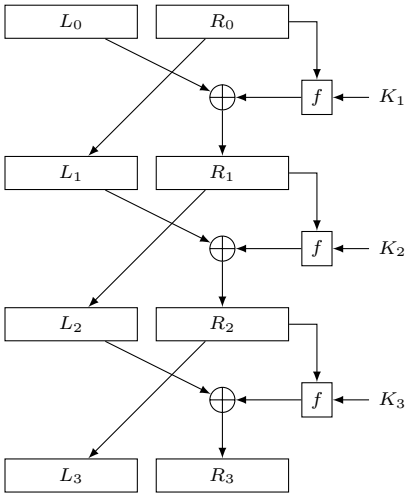
$$f(R_{i-1}, K_i) \equiv P_2 A(P_1 R_{i-1} + K_i) \pmod{28}.$$

Slutligen beskriver konstruktionen av rundnycklarna K_1, K_2 och K_3. Låt $K = (k_1, k_2, k_3)$, där $k_j \in \mathbb{Z}_{28}$, beteckna huvudnyckeln från vilken rundnycklarna genereras. Sätt

$$K_0 \equiv (k_3, k_1 + k_2, k_2, k_2 + k_3, k_1, k_1 + k_3) \pmod{28}$$

och

$$S = (3, 5, 7, 11, 13, 17).$$



Figur 1. De tre rundorna i kryptot.

Då genereras rundnycklarna enligt

$$K_i \equiv \mathrm{LS}_{i+2}(K_i) + S \pmod{28},$$

där $\mathrm{LS}_j(K)$ betecknar ett cykliskt vänster skift j steg av elementen i K.

(a) Låt $\boldsymbol{u} = (1, 2, 3, 4)$ och $\boldsymbol{v} = (1, 2, 3, 4, 5, 6)$. Bestäm $P_1\boldsymbol{u}$ och $P_2\boldsymbol{v}$.

(b) Bestäm rundnycklarna K_1, K_2 och K_3 då huvudnyckeln är $K = (10, 7, 21)$.

(c) Låt R vara det block som motsvarar klartexten `fyra` och låt K_2 vara den andra rundnyckeln från föregående deluppgift. Bestäm $f(R, K_2)$.

(d) Implementera krypteringsfunktionen `VAHF_kryptera(`$\boldsymbol{m}$`, K)`, där indata är dels ett klartextblock $\boldsymbol{m}$ av längd åtta och dels huvudnyckeln K, d v s utgå från en längre klartext delas upp innan anrop av krypteringsfunktionen.

(e) Kryptera klartexten

```
Basically I believe in peace and bashing two bricks together!
```

då huvudnyckeln är $K = (10, 7, 21)$. Funktionen ska returnera kryptogrammet som en textsträng med enbart bokstäver ur $\mathcal{A}_{\mathrm{sve}}$.

(f) Implementera dekrypteringsfunktionen.

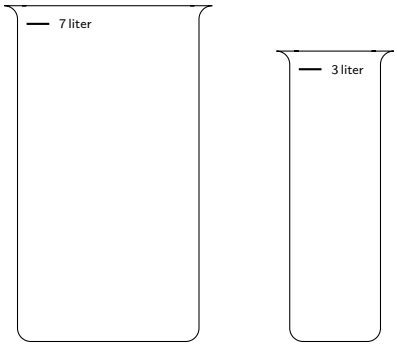
(g) Dekryptera följande kryptogram.

```
VRQHJJBNRJÅFMJFARZPEÖYLCBCXÅIVIKXÖÄMOEMSGAÅDHÄNZLJÄRCEPY
```

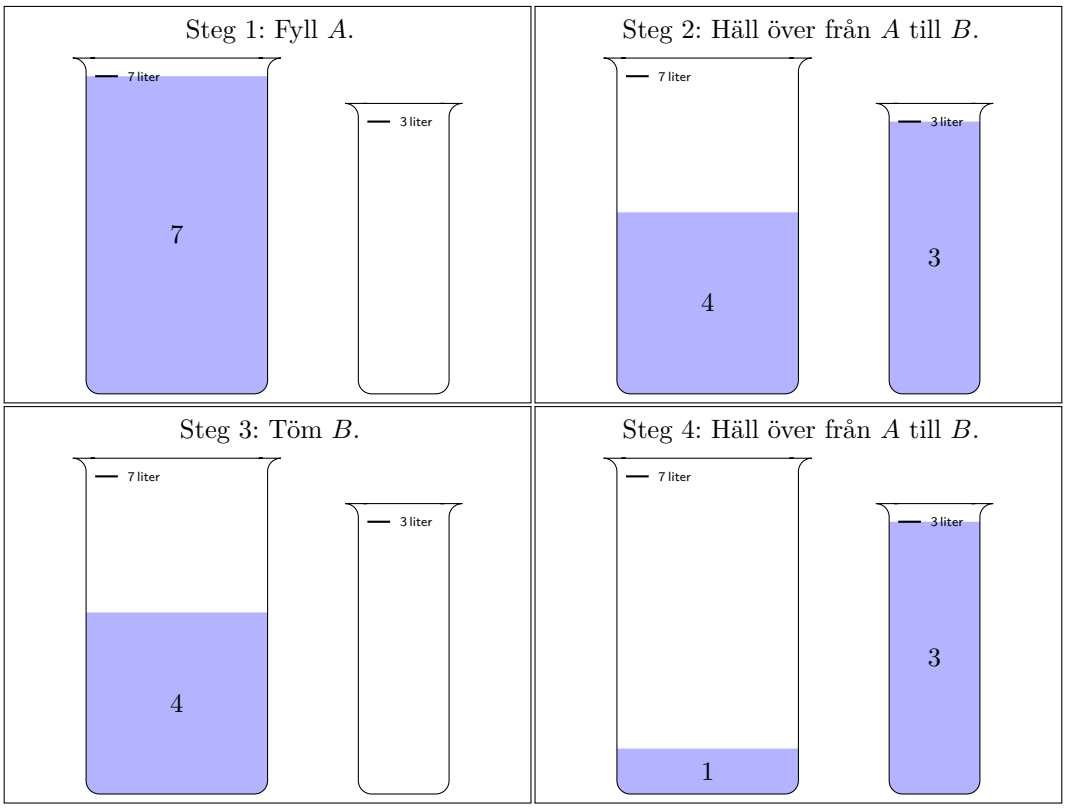
som har erhållits med samma huvudnyckel som ovan.

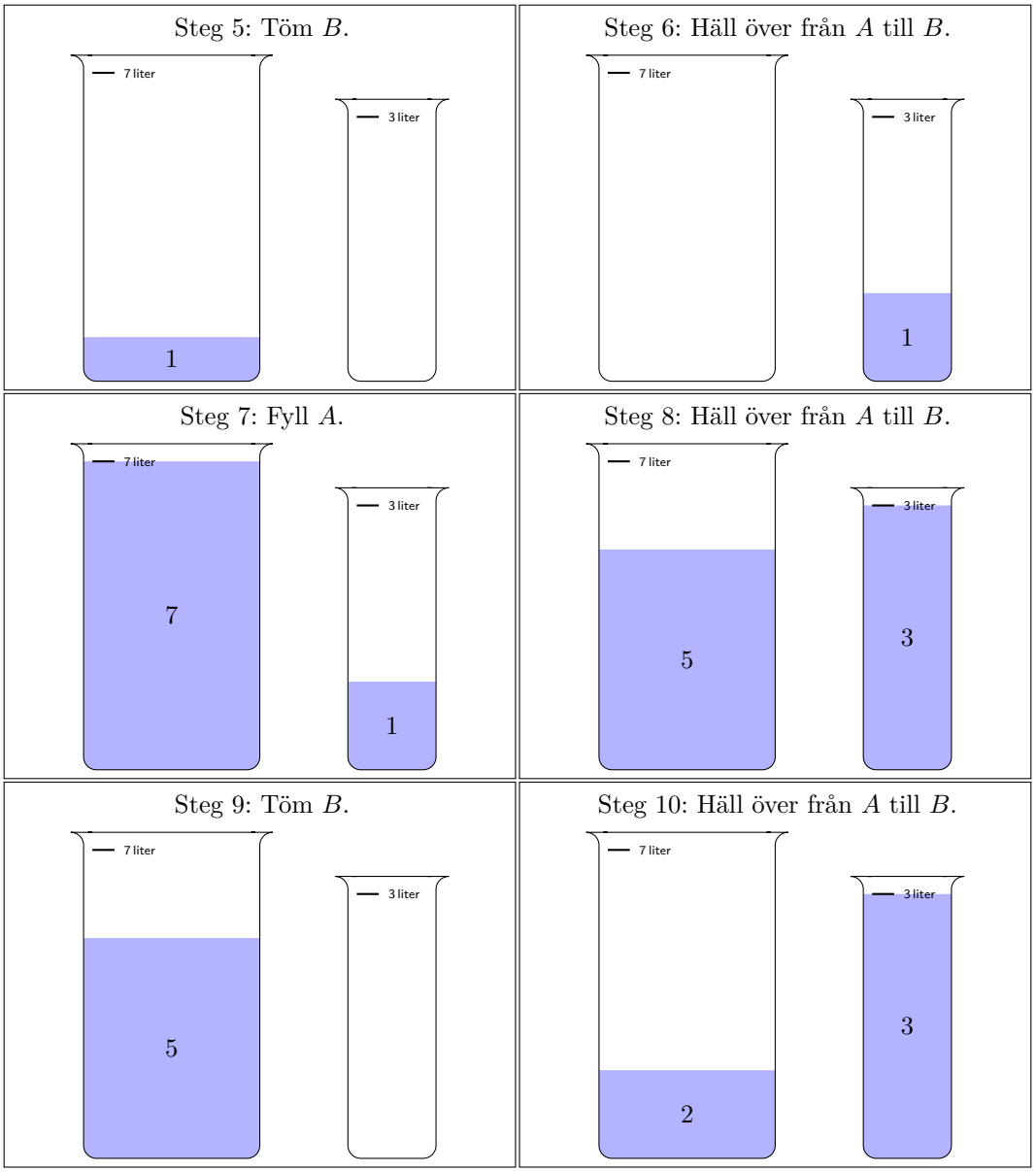
Uppgift 4 (En tillämpning av diofantiska ekvationer)

Antag att vi har två bägare, vilka vi betecknar A respektive B, och antag att de har en markering för volymen 7 respekive 3 liter. Båda bägarna saknar markeringar för 1, 2, . . . , 6 liter. Vidare utgår vi från att vi har tillgång till obegränsat med vatten.



Du önskar mäta upp exakt 2 liter vatten i en av bägarna. En möjlig lsning är följande.





Efter tio steg har vi 2 liter vatten i bägare B. Men hur avgör man om det går att t ex mäta upp 6091 liter med två bägare vars volym är 10 872 respektive 4321 liter? Vidare, om det är möjligt hur bestämmer man stegen för att nå önskad volym?

I exemplet ovan fyllde vi bägare A två gånger och vi fyllde bägare B fyra gånger. Notera att vattnet i B togs från A, d v s

$$7 \cdot 2 - 3 \cdot 4 = 2.$$

Låt a och b beteckna volymen i bägare A respektive B och låt c vara önskad volym. Antag att a, b och c är positiva heltal. Vidare antag att $a > b$, $a > c$ och $b \neq c$. Problemet att mäta upp c liter med hjälp av bägare A och B kan nu omformuleras till att lösa den diofantiska ekvationen

$$ax - by = c,$$

där x och y anger antal gånger som bägare A respektive B ska fyllas. Med andra ord söker vi lösningar där både x och y är positiva.

(a) Programmera en funktion `tvennekannor(`a`, `b`, `c`, visa_steg = True)` som skrier ut en beskrivning om hur man med två bägare A och B, vars volym är a respektive b liter, mäter upp c liter vatten. Funktionen ska returnera listan $[s, x, y]$, där s är antal steg och (x, y) en lösning till $ax - by = c$. Om det är omöjligt ska uppnå den önskade volymen c ska funktionen avbryta, returnera en tom lista och skriva ut följande text.

```
Det går inte att mäta upp c liter med tillgängliga bägare.
```

I t ex fallet $a = 7$, $b = 3$ och $c = 2$ ska utdata se ut som följer.

```
Fyll A.                    (A, B) = (7, 0)
Häll över från A till B.   (A, B) = (4, 3)
Töm B.                     (A, B) = (4, 0)
Häll över från A till B.   (A, B) = (1, 3)
Töm B.                     (A, B) = (1, 0)
Häll över från A till B.   (A, B) = (0, 1)
Fyll A.                    (A, B) = (7, 1)
Häll över från A till B.   (A, B) = (5, 3)
Töm B.                     (A, B) = (5, 0)
Häll över från A till B.   (A, B) = (2, 3)
Önskad volym i bägare A.

Antal steg: 10

En lösning på den diofantiska ekvationen

  7x - 3y = 2

är (x, y) = (2, 4).
```

Om `visa_steg` sätts till `False`, så ska funktionen inte skriv ut något, utan bara returnera en lista.

Du behöver inte implementera någon felhantering för de fall då indata inte uppfyller alla kraven på a, b och c, d v s att a, b och c ska vara positiva heltal sådana att $a > b$, $a > c$ och $b \neq c$. Det lämnas till den som använda funktionen att se till att denne anropar funktionen på rätt sätt.

(b) Bestäm samtliga steg som behövs för att mäta upp 11 liter med en 26 liters bägare och en 17 liters bägare.

(c) Låt $a = 10\,872$ och $c = 6091$. Med vilken eller vilka av $b = 2114$, $b = 4321$ och $b = 6885$ är det möjligt att mäta upp c? Ange antal steg samt x och y i samtliga möjliga fall.

Bilaga

Ledning till uppgift 1

Koda ett hexadecimalt block till ett binärt block

Låt m vara ett en textsträng av längd 16, med hexadecimala siffror. Först splittrar vi upp textsträngen i delsträngar av vardera längd 2.

```
m = '656E746573743132'
H = [m[i:i+2] for i in range(0, 16, 2)];  print(H)
```

$[65, 6E, 74, 65, 73, 74, 31, 32]$

Härnäst konverterar vi varje delsträng till motsvarande heltal decimalt.

```
D = [ZZ('0x' + h) for h in H];  print(D)
```

$[101, 110, 116, 101, 115, 116, 49, 50]$

Varje heltal i listan konverterar vi sedan till en bitsträng.

```
B = [bin(d) for d in D];  print(B)
```

$[0b1100101, 0b1101110, \ldots, 0b110010]$

I nästa steg tar vi bort `0b` i samtliga delsträngar.

```
B = [b[2:] for b in B];  print(B)
```

$[1100101, 1101110, 1110100, 1100101, 1110011, 1110100, 110001, 110010]$

Varje delsträng måste vara åtta bitar lång, eftersom de motsvarar en byte. Därför måste eventuellt nollor läggas till i början av en delsträng. Vidare vänder vi på varje delsträng (med hänsyn till hur funktionen `text_till_block64` är implementerad).

```
B = [b.zfill(8)[::-1] for b in B];  print(B)
```

$[10100110, 01110110, 00101110, 10100110, 11001110, 00101110, 10001100, 01001100]$

Nu kan delsträngarna slås ihop till en textsträng, som då får längdem 64.

```
B = ''.join(B);  print(B)
```

1010011001110110001011101010011011001110001011101000110001001100

Textsträngen konverteras till en lista av heltal, där varje element i listan är en bit, d v s ett element i $\mathbb{Z}_2$.

```
B = [ZZ(b) for b in B];  print(B)
```

$[1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, \ldots 0, 0, 1, 0, 0, 1, 1, 0, 0]$

Ovanstående format fungerar som indata till krypterings- och dekrypteirngsfunktionen – tänk på att `DES_ECB` ska ha en lista av block som indata. Slutligen kan vi konvertera B till en textsträng.

```
block64_till_text([B])  # måste vara en lista av block
```

```
entest12
```

Kodning av klartext för RSA

Kan kan gå tillväga på följande sätt i SageMath för att konvertera från textsträng till heltal och tillbaka enligt förordet i kompendiet.

```
t = 'entest'
A = sve(t);  print(A)
```

$[4, 13, 19, 4, 18, 19]$

Vi behöver addera 1 till var och ett av talen i listan.

```
B = [a + 1 for a in A];  print(B)
```

$[5, 14, 20, 5, 19, 20]$

En lista $[c_k, \ldots, c_{k-1}, \ldots, c_2, c_1, c_0]$ med heltal ur $\mathbb{Z}_{28}$ konverteras till ett heltal enligt

$$m = c_0 + 10^2 c_1 + 10^4 c_2 + \cdots + 10^{2(k-1)} c_{k-1} + 10^{2k} c_k.$$

Det förenklar en del om vi först vänder på listan.

```
C = B[::-1];  print(C)
```

$[20, 19, 5, 20, 14, 5]$

Det är enkelt följa formeln ovan för att bestämma heltalet m.

```
m = sum([10^(2*i) * C[i] for i in range(len(C))]);  print(m)
```

51420051920

Jämför resultatet med B ovan. Det återstår att studera hur man konvertera tillbaka till en textsträng. Först utnyttjar vi divisionsalgoritmen för att bestämma talen $c_0, c_1, c_2, \ldots, c_{k-1}, c_k$.

```
X = []
while m > 0 :
    # m = 100 q + r
    q = m // 100
    r = m % 100
    X = [r] + X
    m = q
print(X)
```

$[5, 14, 20, 5, 19, 20]$

Därefter subtrahera vi 1 från respektive heltal i listan.

```
Y = [x - 1 for x in X];  print(Y)
```

$[4, 13, 19, 4, 18, 19]$

Slutligen kan vi konvertera listan till en textsträng.

```
z = sve(Y);  print(z)
```

```
entest
```

Ledning till uppgift 3

För information om matrisberäkning modulo 28 se avsnitt ”3.3 Hillkrypto” i dokumentet *Kryptering i SageMath*. De kan behöva växla mellan lista och vektor. Konvertering av en lista till en vektor och vice versa:

```
u = [1, 2, 3, 4]          # en lista
v = vector(Zmod(28), u)   # en vektor
list(v)                   # tillbaka till en lista
```