All Exercises

Konstantin Ziegler

March 19, 2013

Contents

1. Classical Cryptography

exe:shift-cipher

Exercise 1 (Shift-Cipher)**.** A secret message M is encrypted with the Shift-Cipher using a secret key K.

```
AZ = AlphabeticStrings()
S = ShiftCryptosystem(AZ)
```

```
M = AZ.encoding('<secret message>')
K = S.random_key()
C = S.enciphering(K, M)
```

The resulting ciphertext C is KYVIVRIVRCJFCVKKVIJFWYZJKFTZTVIFRJNVCC
RJKFYZJZEKZDRKVJFEGIZMRKVRWWRZIJREUZEKYVCRKKVIZWYVREPKYZEXTFEWZ
UVEKZRCKFJRPYVNIFKVZKZETZGYVIKYRKZJSPJFTYREXZEXKYVFIUVIFWKYVCVK
KVIJFWKYVRCGYRSVKKYRKEFKRNFIUTFLCUSVDRUVFLKZWREPFEVNZJYVJKFUVTZ
GYVIKYVJVREUXVKRKKYVZIDVREZEXYVDLJKJLSJKZKLKVKYVWFLIKYCVKKVIFWK
YVRCGYRSVKERDVCPUWFIRREUJFNZKYKYVFKYVIJ.

(a) Retrieve the original message M through exhaustive search.

(b) Can you also recover the secret key K?

Hint: For a candidate key k, you obtain the decryption of C with k using

```
S.deciphering(k, C)
```

```
for k in range(26):
    k, S.deciphering(k, C)
```

The messages is taken from *De Vita Caesarum, Divus Iulius* by Suetonius (c. 69 – after 122 AD). It describes Caesar's cipher.

exe:key-space

Exercise 2 (Exhaustive search of the key space)**.** What is the size of the key space for the following ciphers.

(a) Shift-Cipher

(b) Substitution-Cipher

(c) Affine-Cipher

(d) Vigènere Cipher (with block length m)

(e) Hill Cipher (with block length m)

(f) Permutation Cipher (with block length m)

Which of them is vulnerable to a break by exhaustive search on a common personal computer?

x-of-coincidence

Exercise 3 (coin flips)**.** What is the index of coincidence for a sequence of coin flips produced by a

(a) fair (50/50),

(b) slightly biased (51/49),

(c) heavily biased (90/10) coin?

Friedman provides an index-of-coincidence of - 0.0667 :: English - 0.0738 :: Italian - 0.0762 :: German - 0.0775 :: Spanish - 0.0778 :: French

William F. Friedman, Military Cryptanalysis, Part III, 1939 (revisions by Lambros D. Callimahos as Military Cryptanalytics – published Part I and II, classified Part III)

stitution-cipher

Exercise 4 (Substitution-Cipher)**.** A secret message M is encrypted with the Substitution-Cipher using a secret key K.

```
AZ = AlphabeticStrings()
S = SubstitutionCryptosystem(AZ)

M = AZ.encoding('<secret message>')
K = S.random_key()
C = S.enciphering(K, M)
```

The resulting ciphertext C is XVJIRIXJHHMIXNJZKOEXPEOLLRPAUWXHOXVJHO XPEOLLRPUFJEPZVJKONLJPXHMIXUWXNJKJQMPKJFXWNJIJEKJXOUFPXHMIXNJON LJXWAOLLPUXWXVJVOUFIWAXVJJUJHRTPXVWMXPUEWUYJUPJUEJPXISJRHMIXNJE WHHMUPEONLJOUFKJXOPUONLJTPXVWMXXVJVJLZWATKPXXJUUWXJIOUFEVOUDJON LJWKHWFPAPONLJOXXVJTPLLWAXVJEWKKJIZWUFJUXIPXHMIXNJOZZLPEONLJXWX JLJDKOZVPEEWKKJIZWUFJUEJPXHMIXNJZWKXONLJOUFPXIMIODJOUFAMUEXPWUH MIXUWXKJQMPKJXVJEWUEWMKIJWAIJYJKOLZJWZLJAPUOLLRPXPIUJEJIIOKRDPY JUXVJEPKEMHIXOUEJIXVOXEWHHOUFPXIOZZLPEOXPWUXVOXXVJIRIXJHNJJOIRX WMIJKJQMPKPUDUJPXVJKHJUXOLIXKOPUUWKXVJSUWTLJFDJWAOLWUDIJKPJIWAK MLJIXWWNIJKYJ. The frequency distribution of the ciphertext is given by

```
C.frequency_distribution()
```

and you can obtain the decryption of the text for a guessed key k by

```
k = AZ("DEFGHIJKLMNOPQRSTUVWXYZABC")    # Caesar's cipher
S.deciphering(k, C)
```

(a) Retrieve the original message M by comparing the frequency distribution of the ciphertext with the frequency distribution of an "average" english text given by This is the frequency distribution determined by Beker and Piper (1982). You can compare this with the alternative by Lewand (2000).

(b) Can you also recover the secret key K completely?

include discussion of bigrams

This is from the English translation of Auguste Kerckhoffs's, *La cryptographie militaire*, Journal des sciences militaires, vol. IX, pp. 5–38, Jan. 1883, pp. 161–191, Feb. 1883.

xe:affine-cipher **Exercise 5** (Affine cipher)**.** We want to encrypt a message $x \in \mathbb{Z}_{26}$ with a key $(k_0, k_1) \in \mathbb{Z}_{26}^2$ using the following encryption algorithm enc.

Algorithm: enc
Input: $x \in \mathbb{Z}_{26}$, $(k_0, k_1) \in \mathbb{Z}_{26}^2$
Output: $y \in \mathbb{Z}_{26}$
return $y \leftarrow k_1 \cdot x + k_0$ in $\mathbb{Z}_{26}$

(a) (4 points) Specify a keygeneration algorithm keygen (Input: none; Output: $(k_0, k_1) \in \mathbb{Z}_{26}^2$) and a decryption algorithm dec (Input: $y \in \mathbb{Z}_{26}, (k_0, k_1) \in \mathbb{Z}_{26}^2$; Output: $x^* \in \mathbb{Z}_{26}$) such that the resulting *affine cipher* is

- correct, and
- the keyspace is maximal.

(b) (3 points) What is the size of the keyspace?

item:1 (c) (4 points) Recover the key (k_0, k_1) for an instance of the affine cipher which produces the following message-ciphertext pairs. *Use as few pairs as possible.*

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathrm{enc}_{(k_0,k_1)}(x)$	13	6	25	18	11	4	23	16	9	2	21	14	7	0

x	14	15	16	17	18	19	20	21	22	23	24	25
$\mathrm{enc}_{(k_0,k_1)}(x)$	19	12	5	24	17	10	3	22	15	8	1	20

(d) (3 points) The attack of (c) is a *chosen*-message-attack. What is the expected number of randomly chosen *known* message-ciphertext pairs that a successful key recovery requires?

(e) (4 points) Shannon (1949) suggested to combine cryptosystems as "product ciphers" to obtain systems with higher security. Evaluate the security of the product of *two* affine ciphers.

:vigenere-cipher

Exercise 6 (Vigenère Cipher)**.** A secret message M is encrypted with the Vigenère Cipher using a secret key K of length 6.

```
AZ = AlphabeticStrings()
blocklength = 6
S = VigenereCryptosystem(AZ, blocklength)

M = AZ.encoding('<secret message>')
K = S.random_key()
C = S.enciphering(K, M)
```

The resulting ciphertext C is
VYCEKCDCCBLCHTPNIHQXPPIVARLSLSEICRRGAJ
RTFGHLPCBGJRLXGHGICHMWPXYEIZKTYIBCPFDRHAOLLXVOVZMCMVGFPNBBVYGHI
ORVPPMVGFPNHTUVAGXQAJWHMSOJGHWSXVJDISFKFTTDRIMPVVKJMCTHJVMGXHKT
YAESXVJPGRKJGCMSPUCSMCEFKEESOVLIMVGKPTTHOVLIYCWEBXGGVRLSTFFNMGD
GQEAGRDVFEGTDJPRWXFGRBTMOKCCSLHWUWXLACUCDYHJVKPGMUKYCWOTURNISUF
DRHRGJYCWQKGFTKGCEBDYHJVUPRGQWZGXOMZLVMVGDUTPWNCZTFCTVADGQGILTW
KKKFIASIVLTKONDYIASORRXVONJRGNQVLPTTBFGPDISTKGTLCHJCRKSEPQNLHGD
QIASVICPMAGERXLZKDGIXRKEATKHCZLLTMUWGGLHVYCGXOTVRWKSGXCCXFCCRNI
SUFDHXQTVANLMUKCBVCPTCPEAGERHRGVVKHBBECSSBBIJSRAAGKFDWGCJGCOWUZ
ZAXWPBADGQGRJXGUCDCHLOIVGCTBKELDVSPKRTQHQIGCTTCBCRHJGIGCZQTPNIH
UTRKDKCVYCGFSVYMSLWPNFXVVVYCTQWUKCCVSQWRWXAGJQPZSKJADGQGRJTWTTF
KIASGECBRDTZTPVMUPQIXAUWMGXLCDNAXGRVCRAWPMCGLWQEGCPVKTFHISEZYAX
EWZNBXBVZQGXEWZPTWHQICRHJGIRWXAGJQPZSVISTLSEICRRGAJRTFGYYCGXHJV
KTTBKEEDYHJVKTLGCXCXLQQEATTZGUZNVWRYCGVCFVCIVONKFDNUJZRHXLKJRTG
QGZQCHHJZBSXBCEBIASGECBRWURQHNAGURDAOXVYCRGRVAXTZGHSXIAGERCXQGJ

QPKMVFGCMSTTCEMOPUPTVCTURWXHTRLHFWVKCSLWIEYAPSEFLHBRGIMCEMVYCIA
WTURNISEFLRXONDCCMGAJRTFOTVNGBACIGARORJWRACNFEXVONGPDUZGDYCWDTZ
TPVMUPQIXAURRTVVPFJDZWERJDGSUVADGRNPRWXHTVYIFSPKGHEWOZRTWHQKFTV
OUVMUWWUTPTMSKEDDKACKGDGKJVPTMVGDCHLOIVRDUSGEAXIVGICSVCPJGHMGQW
YHXEWVLRXCHUGHVFGKCHRADFJHXOEYAWHGGEDGHACWGCBHGJCIMVGJCHRADFJHF
OASCAXHVVPHBBCCYCZICXCLHFFJMUTZCEEJTUGRKEEWVLBTESXVJHHTCHSPGHKQ
CSLDGVAWHFXZBTHGKXLPESVTZJMHJVKPBBGDNWTGKJYCWHJZLZBBIYYHUSGEADG
QGILTWKKKFIASERQTHTNVRIXFUKFTIORVPXLRKMGSXRKERDMVTVCETFVJRWXACZ
LGXGWCRHPWNCLDPPGSPXXTNPQJFACIGOXRVYCUBFUKNPKHFVYALKKKFIASDRQXV
ACKFTFOVZAPEGVISRMITVMULSEICRRGAJRTFGCJGCVCODSCBQCKGDGHJVMGRONR
LVNOIVGHVCPJGSXFGURDUSTVNGXGGERTWPARQIHQJRQIBQRIMRXGUNFXVVRIMSN
QGJYSBGEICIXGGHSTGQGFDHRADFJHBBCTADKRCEATPWVYQDFSUPQIXAQWNGHPCS
GABHKVQPLGQTGPMSFNGIAONRLVNOIVRWXFGZQPVSTKYXGDCIYBXHGIBLAWEYUTV
ONCRWXFGUSCWOPTWDYHJVJPGUWRETWAGRQJKSUZLPLSPJCWHKOLAWTHGORXGHJV
JPGUWRETVOPSCGXRWTCSBBNVLVMVYZRWHIVCMHBBIRLNBBHFPBTHKFLPLOUZKEE
SGOYBIZGJGCVSWRJLTMUWMAECYJOXGSPXJXLVYFPSLHJVSBTMDVMBBHVVBLBHJF
SIECUJADGGKUCGTPNVPTWIEKGDGGCICEHGUZZAXWPVLVEWUYBJXHQKFTLHCKGHM
WERJHMFWTRJKSQWRWXZCEEJTUGKFTAWIYDGXEWVLRBSUFDRXFVRGCESVKCGLCTN
MGWGGKAGXRWEBPGQAZQDYQGERGTZKDNDKHCEATBBVYCHMIFPMULSEICRRGAJRTF
GCJCRKSEPQNLHGDGHWSHZLTWODJRGTQVCWPLOUVRDYHTRLHYCTDYIBCPJMUHBGJ
NPVSVYCHXHQWNDLGKSJTFSUJYVXGKERDTGGTMCWGRRATMVGJCIHTRFQHBPNVAGR
DVFEGTAUVYRADCIRXVINRPIKOPJDDKACKGDGCHKFTLSVTMGKSUGMCWGVFCCVWRY
CGBBINGIAORRPIBQWCYGDSAKFTMFCEQUHFORRXHBURPTLIRGMHXRTVTTKGKSJTG
CPJGCZINRPHHHJRRJGWSLCSXQKGFTKWPXGHICUJGQESYYCCMVGBCNBGMEMLGSCT
FZXMCEBIASTVDDKSGRAWMFCEQUHFORRXHBKJYHLIOVBIHVCMCPGORIGDKWRIMQT
PKCGIROUJMRBOVVBLBHJZRIASRIMQTPKCGIRCHTFDHGKEEIAOVBCNLWOZJPKZAV
YRADQJQXUZGDCHLOIVGHTGULKTWHQYYKXOPRQHHQKRRTWORIGDKWRIMQTPKCGIR
RGKCGFWPVBQRHJVSCWSTCWXGUUKMRAOUKGRIFQTCHLHJVQTIFQSYQBZKKGTLTQI
RWXJCIGDNGMVWHTBFDCHLOIVQPKSCTRJTZNPRWXSPVKNVFAGRPGONPQILORIGDK
WRIMQTPKCGIBSUWMGMVGTFDBQGJGCJIGJRXHBCEBGXDTVQTGHJZQPIFKFPXDBQN
JTWUGFDIASUZRJTHKFLIHIUVRWXGAJRTFOMVWXLTKIQILSNVAIXRCEBHXBVKMIA
STVATBJKEEEHWPKRWXQJFGRXCHRITRRGKCGFWPVQPIOTKGRNZCIRGTBUWMGFOVZ
MCBBVYCHXHHFPBBBIKFTLMUKCBMVGEYBXGURETBGUVJTVHGUYCWHJVNPKHKTSAT
FVIYCLTQIKPMWQEADKFGJNDGRKEEIHHJVQTESEKCSDSARNEEWGURDMVKJKTLGCX
CIHDTFBJVSCTPNIHQXPPFHJZQRKMRKMVKOOZQIKOPJKXMHGURDMVGICRXWXZLVI
CKERQROEYYCGSNRLSFOASCXGHGIATIHGUZNMVGVLTFMCKRWXFGTCXOWPXCCWHJV
GCOSTJCDYHJVNPKHKTSATFVIYCLTQIKPMWQEGHTDRCGTWHQKFTVFAGRDZFCDRDK
SEFTTKHJVMGBUKEYAFSUJYVXWHKFTXBGDWXGHGIATIHUKFTVFAGRDZFCDFTVOPT
YAVINRRTYFQDGIMVGRNDLHGIGDKWRIMQTPKCGIBSUFDIASXRPXHIUGMHLWDCCBX
GURETLOPUITRGYYGRAAKXFIAOXVNGHRWTCSMVKJAGRDVFEGTAVYGHLSVFDPICUK
CGBCTZNGHPCSGABHKVQRHBUKGINHGJFXLYPFUAXRIVMUMVGBCNTBFDCHLOIVYUM

```
STKFTBBVVPRXDVZMCDBQNJTWUGZQIAIUZBTGHKWGTWKKKFPLSVFDEKCRFQXMWQE
QWTJKEEPLGQTGPMSFGPDUODZJXMWGJRWXQCCAJEOVZMCHTVYCPICUKCGBCTZNGH
PCSGABHKVQXLHJVETGSTRJXSSFGPDUZGDMUVFAGRPGONPQXL.
```

(a) Retrieve the original message M.

(b) Can you also recover the secret key K?

Hint: You can address all letters in the ciphertext at positions congruent $i \bmod 6$ for $0 \leq i < 6$ and its frequency distribution by

```
for i in range(6):
    B = C[i::6]
    B.frequency_distribution()
```

Shannon (1949), Communication Theory of Secrecy Systems

xe:random-matrix **Exercise 7** (a random matrix is invertible)**.** No exercise yet.

Conjecture for binary matrices. Verify asymptotically.

Formulate an analogue conjecture over $\mathbb{F}_q$ and test it.

exe:hill-cipher **Exercise 8** (Hill cipher)**.** No exercise yet, just the Sage code for experiments

```
blocklength = 6
G = SymmetricGroup(blocklength*blocklength)
S = [i+5*j for i in range(1,6) for j in range(5)]
G(S)    # cycle notation
```

e:product-cipher **Exercise 9** (product cipher)**.** TODO

Build a product-cipher from classical crypto. Does that help? (No.) Why not? (Linearity!)

What if you employ the non-liner Substitution Cipher?

Why does the product-cipher design strengthen AES? Non-linear SubBytes.

eat-expectations **Exercise 10** (great expectations)**.** (a) (expected time 'til occurrence) The least common letter in the English alphabet is 'z' with $p_z = 0.074$. What is the expected length of text (just letters, no spaces) until 'z' occurred at least once? What is the expected length until each letter occurred once?

(b) (When does the law of big numbers begin?) For which length of text do you expect 'e' to be the most common letter with probability at least 95%? For which length do you expect 't' to be the 2nd most common with probabililty at least 95%? Run experiments.

(c) (distinguishing "close" frequencies) The last exercise is about identifying the "dominant" elements. But, how hard does it get, if the frequencies are close to each other? Instead of investigating letters, with small frequencies we turn to a more practical example.

Let $\varepsilon > 0$, $f = 0.5 + \varepsilon$ and $m = 0.5 - \varepsilon$ be the proportion of women and men in a large population, respectively. Assume a person is sampled according to this distribution. What is your best guess for its gender and what is your probability of success. Assume now, you are given a database with clear (distinct) family names and encrypted gender for each entry. How large does the database have to be (depending on ε) such that the most common entry is the encryption of 'f' with probability at least 95%? Run experiments.

??

This is the Coupon collector's problem. For uniform letter distribution this would be $n \cdot H_n$, where $H_n = 1/1 + 1/2 + 1/3 + \cdots + 1/n$ is the n-th Harmonic number. Asymptotically, this is

$$nH_n = n\ln(n) + \gamma(n) + 1/2 + o(1) \approx 100(\text{for} n = 26) \tag{eq:1}$$

2. AES

exe:aes-invert **Exercise 11** (The modified last round of AES)**.** (7 points)

The encryption of a message x under 10-round AES with a key k can be described as follows.

```
x = aes.add_round_key(x, k)
for i in range(1,11):
    x = aes.sub_bytes(x)
```

```
        x = aes.shift_rows(x)
        if i < 10:
            x = aes.mix_columns(x)
        k = aes.key_schedule(k,i)    # determine round-key
        x = aes.add_round_key(x,k)
    y = x
```

Show that the decryption can be achieved with the same structure. In other words, make suitable changes to the key-schedule, the S-box, the shifting operation on the rows, and the mixing operation on the columns to decrypt y using the key k *with the same sequence of (modified) operations.*

SPN-generalization, Stinson 3.1

Feistel, Stinson 3.2

xe:aes-implement

Exercise 12 (your implementation of baby-AES)**.** (a) Turn the pseudo-code of the previous exercise into a function $\text{enc}(M, K, i)$ which computes the state of AES after round i of our 3-round AES given message M and key K. In other words $\text{enc}(M, K, 0)$ should return $M + K^0$ and $\text{enc}(M, K, 3)$ should return $\text{aes}(M, K)$.

(b) For later use, write a function $\text{declast}(C, K^n)$ which decrypts the last round, i.e. $\textit{declast}(C, K^n)$ should return the state before entering the last round, given the ciphertext C and the last round-key K^n.

Hint: The inverse of MixColumns and Shiftrows are particularly easy to describe for our baby-AES. Hint: What happens of you execute each of the operations twice?

Hint: The inverse of an S-box is again an S-box. You can specify an S-box in Sage explicitly by

```
mySbox = mq.SBox(14, 13, 4, 12, 3, 2, 0, 6, 15, 8, 7, 1, 11, 9, 5, 10)
```

3. Differential Cryptanalysis

exe:diff-attack

Exercise 13 (Differential cryptanalysis)**.** In the lecture, we found a differential trail through the first two rounds of baby-AES with propagation ratio 1/64. For the corresponding differential attack, we required 192 pairs of plaintext-ciphertext pairs with corresponding input difference.

For this exercise, the S-box of baby-AES is replaced with the following new 4-bit S-box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	E	2	1	3	D	9	0	6	F	4	5	A	8	C	7	B

We call this the new baby-AES.

cor:1 (a) (3 points) Compute the output difference distribution for input xor $x' = 0001$. [Hint: Eight xors suffice.]

cor:2 (b) (4 points) The difference distribution table of the new S-box is displayed below, but the first three rows are missing. Complete the table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3	0	2	0	4	0	2	0	0	0	2	0	4	0	2	0	0
4	0	4	2	2	0	2	0	2	2	0	0	2	0	0	0	0
5	0	0	0	4	0	0	0	4	0	0	0	0	2	2	2	2
6	0	0	0	0	2	0	2	0	2	0	2	0	2	2	2	2
7	0	0	4	2	2	0	0	0	4	2	0	0	0	0	2	0
8	0	2	0	0	2	4	2	2	0	2	0	0	0	2	0	0
9	0	6	0	0	0	0	2	0	0	0	2	4	0	2	0	0
A	0	0	2	0	0	0	0	2	4	0	4	2	0	0	2	0
B	0	0	0	0	2	2	2	2	0	0	0	0	4	0	4	0
C	0	0	2	0	0	2	4	0	2	0	0	0	2	2	2	0
D	0	0	2	2	2	0	2	0	0	2	6	0	0	0	0	0
E	0	0	2	2	0	0	0	0	2	6	0	0	0	0	0	4
F	0	0	0	0	2	4	0	2	0	2	0	2	2	2	0	0

(c) Use Sage to compute the difference distribution matrix for this S-box and check that results from (a) and (b).

(d) (1 point) What is the maximal difference probability?

cor:3 (e) (3 points) Give an upper bound for the propagation ratio of a differential trail through the first two rounds of our new baby-AES.

(f) Find a differential trail through the first two rounds of our new baby-AES whose propagation ratio achieves the upper bound of (e).

(g) (2 points) How many pairs of plaintext-ciphertext pairs will you request for a differential attack against our new baby-AES using a trail whose propagation ratio matches the upper bound obtained in (e). Use the same implicit constant as we used for the attack on the original baby-AES described at the beginning.

X	Y	Y'
0000	1110	0101
1111	1011	
0001	0010	0101
1110	0111	
0010	0001	1101
1101	1100	
0011	0011	1011
1100	1000	
0100	1101	0111
1011	1010	
0101	1001	1100
1010	0101	
0110	0000	0100
1001	0100	
0111	0110	1001
1000	1111	

MISSING: difference distribution table.

??

exe:diff-const

Exercise 14. You visit a casino with 2^k lotteries which have a probability of winning of $1/2^\ell$ each. One of them is broken though and has a probability of winning of $p + 1/2^\ell$ with $p > 0$.

We run the following experiment to find the "lucky" machine

1. Run each lottery N times and record the number of "wins".
2. We call the set of machines with the most wins W

3. The experiment is *successful* if the "lucky" machine is an element of W, and *uniquely successful* if the "lucky" machine is the unique element of W.

Determine by experiment the answer to the following questions for $k = \ell = 8$ and $p = 1/64$.

for a filtered set, $p = 1/4$.

1. For which size of N do you expect the experiment to be successful.
2. For which size of N do you expect the experiment to be uniquely successful.

```
def gambling(N):
    k = 8
    l = k
    nlotteries = 2^k
    pwin = 1/2^l
    biasbroken = 1/64
    indexbroken = floor(random() * nlotteries)

    counters = [0] * nlotteries
    for lottery in range(nlotteries):
        prob = pwin
        if lottery == indexbroken:
            prob += biasbroken
        for i in range(N):
            if random() < prob:
                counters[lottery] += 1
    winners = set()
    maxcounter = max(counters)
    for lottery in range(nlotteries):
        if counters[lottery] == maxcounter:
            winners.add(lottery)
    successful = indexbroken in winners
    uniquelysuccessful = successful and len(winners) == 1
    return successful, uniquelysuccessful

# Try different N
for N in (100, 500, 1000):
    s = 0
```

```
        us = 0
        for i in range(100):
            success, unique = gambling(N)
            if success:
                s += 1
            if unique:
                us += 1
        print '*** N = %d ***' % N
        print 'Successful in %d out of 100' % s
        print 'Uniquely so in %d out of 100' % us


################### Output ############################

*** N = 100 ***
Successful in 34 out of 100
Uniquely so in 20 out of 100
*** N = 500 ***
Successful in 89 out of 100
Uniquely so in 83 out of 100
*** N = 1000 ***
Successful in 99 out of 100
Uniquely so in 98 out of 100
```

Alternative solution

```
    #exercise 2.2 (1)
    import random
    def lottery(N,k,l,p):
        wins = [0]*(2**k)
        prob=1/(2**l)
        broProb=p+prob
        for i in range(N):
            for j in range (2**k):
                rand = random.random()
                if (j != (2**k)-1):
                    if (rand <= prob):
                        wins[j] +=1
                else:
                    if (rand <= broProb):
```

```
                        wins[j] +=1
        return wins

    lottery ( 1000, 8, 8, (1/64))
```

```
[2, 3, 4, 5, 5, 2, 3, 5, 1, 2, 3, 5, 3, 3, 5, 3, 0, 4, 4, 3, 5, 3, 5, 7,
2, 2, 5, 1, 2, 5, 4, 3, 3, 2, 4, 5, 1, 3, 3, 2, 5, 6, 2, 5, 1, 4, 2, 3,
5, 4, 1, 4, 4, 1, 5, 0, 3, 1, 5, 5, 7, 5, 4, 5, 9, 1, 4, 3, 2, 4, 4, 5,
5, 6, 5, 3, 5, 3, 8, 2, 5, 8, 8, 2, 6, 6, 3, 2, 4, 7, 3, 3, 9, 1, 5, 6,
2, 4, 4, 4, 5, 7, 3, 4, 6, 3, 4, 6, 5, 2, 6, 2, 4, 2, 3, 8, 2, 3, 2, 2,
5, 5, 9, 6, 0, 4, 7, 3, 4, 6, 5, 4, 4, 3, 4, 6, 3, 5, 7, 4, 5, 3, 1, 4,
4, 4, 2, 3, 2, 7, 2, 2, 2, 6, 2, 5, 4, 2, 5, 6, 1, 3, 3, 4, 5, 3, 4, 2,
1, 3, 9, 6, 0, 3, 7, 4, 5, 3, 6, 2, 5, 4, 5, 4, 3, 2, 5, 3, 1, 6, 0, 5,
4, 4, 5, 2, 7, 1, 5, 2, 5, 5, 4, 4, 1, 8, 5, 2, 5, 4, 5, 5, 7, 3, 4, 2,
4, 2, 7, 4, 4, 4, 3, 2, 4, 2, 4, 0, 5, 7, 1, 6, 2, 5, 0, 5, 3, 4, 1, 5,
4, 6, 3, 0, 3, 4, 4, 2, 5, 2, 1, 3, 3, 3, 4, 25]
```

```
    #exercise 2.2 (2)
    import random
    def lottery(N,k,l,p):
        wins = [0]*(2**k)
        prob=1/(2**l)
        broProb=p+prob
        for i in range(N):
            for j in range (2**k):
                rand = random.random()
                if (j != (2**k)-1):
                    if (rand <= prob):
                        wins[j] +=1
                else:
                    if (rand <= broProb):
                        wins[j] +=1
        W = max (wins)
        wins = filter(lambda x:x==W, wins)
        return wins

    lottery ( 500, 8, 8, (1/64))
```

```
[10]
```

```
#exercise 2.3 (i)
import random
def lottery1(N,k,l,p):
    wins = [0]*(2**k)
    prob=1/(2**l)
    broProb=p+prob
    for i in range(N):
        for j in range (2**k):
            rand = random.random()
            if (j != (2**k)-1):
                if (rand <= prob):
                    wins[j] +=1
            else:
                if (rand <= broProb):
                    wins[j] +=1
    W = max (wins)
    lucky = wins[(2**k)-1]
    wins = filter(lambda x:x==W, wins)
    if (lucky == W):
        return 1
    else:
        return 0

def findN1():
    success = 0
    N=2
    while (success < 19):
        success = 0
        N=N*2
        for i in range (20):
            success += lottery1 ( N, 8, 8, (1/64))
    return N


#exercise 2.3 (ii)
import random
def lottery(N,k,l,p):
    wins = [0]*(2**k)
    prob=1/(2**l)
    broProb=p+prob
    for i in range(N):
```

```
            for j in range (2**k):
                rand = random.random()
                if (j != (2**k)-1):
                    if (rand <= prob):
                        wins[j] +=1
                else:
                    if (rand <= broProb):
                        wins[j] +=1
        W = max (wins)
        lucky = wins[(2**k)-1]
        wins = filter(lambda x:x==W, wins)
        if (lucky == W):
            if len(wins) == 1:
                return 1
        return 0

    def findN():
        success = 0
        N=8
        while (success <= 99):
            success = 0
            N=N*2
            print N
            for i in range (100):
                success += lottery ( N, 8, 8, (1/64))
        return N

    findN1()
```

```
1024
```

exe:average-S **Exercise 15** (the average S-box)**.** For the following S-boxes on $\mathbb{F}_{16}$ draw the difference distribution matrix and find the maximal difference probability:

1. identity id,

```
S = mq.SBox(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)    # identity
S.difference_distribution_matrix()
S.maximal_difference_probability()
```

2. affine linear transformation $x \mapsto (a^3 + a^2 + 1) \cdot x + (a^2 + a)$,

3. (extended) inverse

$$\mathrm{inv}(x) = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{else,} \end{cases}$$

4. baby-AES S-box.

5. Plot the distribution of the maximal difference probability of 1 000 randomly chosen S-boxes.

```
Erst mal alles bis auf die (v):

################## Code ####################

values = list(aes.base_ring())
values.sort()

def printstuff(sbox):
    print 'Max diff prob: %f' % sbox.maximal_difference_probability()
    print sbox.difference_distribution_matrix()
    print

print '*** Identity ***'
identity = mq.SBox(values)
printstuff(identity)

print '*** Affine linear ***'
affinelinear = mq.SBox([(a^3 + a^2 + 1) * x + (a^2 + a) for x in values])
printstuff(affinelinear)

print '*** Extended inverse ***'
sboxout = []
for x in values:
    if x == 0:
        sboxout.append(0)
    else:
        sboxout.append(1/x)
inverse = mq.SBox(sboxout)
printstuff(inverse)
```

```
print '*** Baby-AES S-box ***'
printstuff(aes.sbox())

################## Output ####################

*** Identity ***
Max diff prob: 1.000000
[16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0 16  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0 16  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0 16  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 16]

*** Affine linear ***
Max diff prob: 1.000000
[16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  0]
[ 0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0]
[ 0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0]
[ 0 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0 16  0  0  0]
[ 0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0]
[ 0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0]
[ 0  0 16  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 16]
[ 0  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0]
[ 0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0]
[ 0  0  0 16  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0]
```

```
[ 0  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0]
[ 0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0]

*** Extended inverse ***
Max diff prob: 0.250000
[16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  4  0  0  0  0  2  2  0  2  0  2  0  2  2  0]
[ 0  0  0  2  0  0  0  2  0  4  2  0  2  2  0  2]
[ 0  0  2  0  0  0  0  2  2  0  2  2  2  0  4  0]
[ 0  0  0  0  0  2  2  0  2  0  2  0  0  4  2  2]
[ 0  0  0  0  2  0  2  0  2  2  0  4  2  0  0  2]
[ 0  2  0  0  2  2  2  4  0  0  2  0  2  0  0  0]
[ 0  2  2  2  0  0  4  2  2  0  0  0  0  0  0  2]
[ 0  0  0  2  2  2  0  2  0  0  0  2  0  0  2  4]
[ 0  2  4  0  0  2  0  0  0  2  0  0  2  0  2  2]
[ 0  0  2  2  2  0  2  0  0  0  0  0  4  2  2  0]
[ 0  2  0  2  0  4  0  0  2  0  0  2  2  2  0  0]
[ 0  0  2  2  0  2  2  0  0  2  4  2  0  0  0  0]
[ 0  2  2  0  4  0  0  0  0  0  2  2  0  2  0  2]
[ 0  2  0  4  2  0  0  0  2  2  2  0  0  0  2  0]
[ 0  0  2  0  2  2  0  2  4  2  0  0  0  2  0  0]

*** Baby-AES S-box ***
Max diff prob: 0.250000
[16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  2  2  2  2  0  0  0  2  0  0  0  2  4  0  0]
[ 0  2  0  4  2  2  2  0  0  2  0  0  0  0  0  2]
[ 0  2  4  0  0  2  0  0  2  2  0  2  0  0  2  0]
[ 0  0  2  0  4  2  0  0  0  0  2  0  2  0  2  2]
[ 0  0  0  2  0  0  0  2  4  2  0  0  2  0  2  2]
[ 0  4  0  0  0  2  0  2  0  2  2  0  2  2  0  0]
[ 0  2  0  0  0  0  2  0  0  0  0  2  4  2  2  2]
[ 0  2  2  0  0  0  2  2  2  0  2  0  0  0  0  4]
[ 0  0  2  2  0  0  0  0  0  2  2  4  0  2  0  2]
[ 0  0  2  0  2  0  2  2  0  4  0  2  2  0  0  0]
[ 0  0  0  0  2  0  2  0  2  2  4  0  0  2  2  0]
[ 0  0  0  2  0  4  2  0  2  0  2  2  2  0  0  0]
[ 0  0  0  0  2  2  0  4  2  0  0  2  0  2  0  2]
[ 0  0  2  2  0  2  4  2  0  0  0  0  0  2  2  0]
[ 0  2  0  2  2  0  0  2  0  0  2  2  0  0  4  0]
```

```
################################################
################################################

Jetzt die (v):

#################### Code ##########################

def plotmaxdiffdist(bits):
    freqs = [0] * (2^bits + 1)
    for i in range(100):
        values = range(2^bits)
        shuffle(values)
        s = mq.SBox(values)
        freqs[s.maximal_difference_probability_absolute()] += 1
    points = point([(i, freqs[i]) for i in range(len(freqs))])
    points.plot().show()

for bits in (4, 2, 8):
    print '%d bits' % bits
    plotmaxdiffdist(bits)

#################### Output ########################

nicht in ASCII darstellbar
```

4. Linear Cryptanalysis

exe:bias

Exercise 16 (independent random variables)**.** (3 points) Suppose that X_1, X_2, and X_3 are independent discrete random variables defined on the set $\{0, 1\}$. Let ε_i denote the bias of X_i, for $i = 1, 2, 3$. Under which conditions on ε_i are $X_1 \oplus X_2$ and $X_2 \oplus X_3$ independent? (Recall, that in the lecture, we saw that this is in general *not* the case.)

By the piling-up lemma we know the bias of $X_1 + X_2$ to be $2\epsilon_1\epsilon_2$ and analogously for $X_2 + X_3$. If the two random variables were independent, then we could use the piling-up lemma to find the bias of their sum $(X_1 + X_2) + (X_2 + X_3) = X_1 + X_3$: It would be $2\epsilon_{1,2}\epsilon_{2,3} = 8\epsilon_1\epsilon_2^2\epsilon_3$. On the other hand, we know that the real bias of the sum $(X_1 + X_2) + (X_2 + X_3) = X_1 + X_3$

is $2\epsilon_1\epsilon_3$. So we conclude that the two are independent if the two biases are equal, i.e. $8\epsilon_1\epsilon_2^2\epsilon_3 = 2\epsilon_1\epsilon_3$, which means that $\epsilon_1 = 0$ or $\epsilon_3 = 0$ or $\epsilon_2 = \pm\frac{1}{2}$.

exe:lin-attack **Exercise 17** (Linear cryptanalysis)**.** In the lecture, we found a linear approximation through the first two rounds of baby-AES with bias 1/32. For the corresponding linear attack, we requested 2048 randomly chosen plaintext-ciphertext pairs.

For this exercise, the S-box of baby-AES is replaced with the following new 4-bit S-box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	8	4	2	1	C	6	3	D	A	5	E	7	F	B	9	0

We call this the new baby-AES.

cor:1 (a) (3 points) Compute the bias of the random variable $X_0 \oplus Y_3 \oplus Y_2 \oplus Y_1$.

cor:2 (b) (6 points) The linear approximation table of the new S-box is displayed below, but the first two rows are missing. Complete the table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	0	2	0	2	-2	0	-2	0	-2	0	2	-4	-4	-2	0	2
3	0	0	2	2	0	4	2	-2	-2	-2	0	0	2	-2	4	0
4	0	2	0	-2	0	2	0	-2	2	-4	2	0	-2	0	-2	-4
5	0	4	-2	-2	2	2	0	4	-2	2	0	0	0	0	2	-2
6	0	0	4	0	2	2	-2	2	0	0	0	4	-2	-2	-2	2
7	0	-2	-2	0	4	-2	2	0	0	-2	-2	0	-4	-2	2	0
8	0	2	2	0	0	-2	-2	0	2	0	-4	-2	2	-4	0	-2
9	0	0	0	4	2	2	-2	2	2	-2	-2	-2	0	4	0	0
A	0	4	2	2	-2	-2	4	0	0	0	-2	2	-2	2	0	0
B	0	-2	4	-2	0	-2	0	2	-4	-2	0	-2	0	2	0	-2
C	0	0	2	2	4	0	2	-2	0	4	2	-2	0	0	-2	-2
D	0	-2	0	-2	-2	4	2	0	0	2	-4	-2	-2	0	-2	0
E	0	-2	-2	4	-2	0	0	2	-2	0	0	2	0	-2	-2	-4
F	0	0	0	0	0	0	4	4	2	-2	2	-2	2	-2	-2	2

(c) Use Sage to compute the linear approximation matrix for this S-box and check that results from (a) and (b).

(d) (1 point) What is the maximal absolute value of a bias?

cos:3 (e) (4 points) The maximal absolute value for nonzero random variables listed in the linear approximation table is 4. Give an upper bound for the absolute value of the bias of a linear approximation through the first two rounds of our new baby-AES. [Hint: Piling-Up Lemma.]

(f) Find a linear approximation through the first two rounds of our new baby-AES whose bias achieves the upper bound of (e).

(g) (2 points) How many plaintext-ciphertext pairs will you request for a linear attack against our new baby-AES using a linear approximation whose bias matches the upper bound obtained in (e). Use the same implicit constant as we used for the attack on the original baby-AES described above.

(h) (3 points) Compare the amount and the type of encrypted information an attacker requires for differential and linear attacks, respectively.

(i) (4 points) The differential and linear attack presented in the lecture do not recover the complete secret key. State precisely their actual output and argue why this is in many situations still considered a "break". [Hint: The key schedule of (baby-)AES is invertible.]

```
sage: S = mq.SBox(8,4,2,1,0xC,6,3,0xD,0xA,5,0xE,7, 0xF,0xB,9,0)
sage: S.linear_approximation_matrix()
[ 8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[ 0  2 -2  0  2  0  0 -2 -4 -2 -2  0  2  0 -4  2]
[ 0  2  0  2 -2  0 -2  0 -2  0  2 -4 -4 -2  0  2]
[ 0  0  2  2  0  4  2 -2 -2 -2  0  0  2 -2  4  0]
[ 0  2  0 -2  0  2  0 -2  2 -4  2  0 -2  0 -2 -4]
[ 0  4 -2 -2  2  2  0  4 -2  2  0  0  0  0  2 -2]
[ 0  0  4  0  2  2 -2  2  0  0  0  4 -2 -2 -2  2]
[ 0 -2 -2  0  4 -2  2  0  0 -2 -2  0 -4 -2  2  0]
[ 0  2  2  0  0 -2 -2  0  2  0 -4 -2  2 -4  0 -2]
[ 0  0  0  4  2  2 -2  2  2 -2 -2 -2  0  4  0  0]
[ 0  4  2  2 -2 -2  4  0  0  0 -2  2 -2  2  0  0]
[ 0 -2  4 -2  0 -2  0  2 -4 -2  0 -2  0  2  0 -2]
[ 0  0  2  2  4  0  2 -2  0  4  2 -2  0  0 -2 -2]
```

```
[ 0 -2  0 -2 -2  4  2  0  0  2 -4 -2 -2  0 -2  0]
[ 0 -2 -2  4 -2  0  0  2 -2  0  0  2  0 -2 -2 -4]
[ 0  0  0  0  0  0  4  4  2 -2  2 -2  2 -2 -2  2]
```

in-approx-matrix

Exercise 18. Let $S\colon \{0,1\}^m \to \{0,1\}^n$ be an S-box. Prove the following properties of the linear approximation matrix.

(a) $NL(0,0) = 2^m, NL(*,0) = 2^{m-1}$

(b) $colsum = 2^{2m-1} \pm 2^{m-1}$

(c) $totalsum 2^{n+2m-1} or 2^{n+2m-1} + 2^{n+m-1}$

An S-box is *balanced*, if $\#S^{-1}(y) = 2^{n-m}$ for all $y \in \{0,1\}^n$.

(a) If S is balanced, then $NL(0,*) = 2^{m-1}$.

(b) If S is balanced, then $rowsum = 2^{m+n-1} - 2^{m-1} + i2^n$, where i is an int $0 \le i \le 2^{m-n}$.

exe:bias2

Exercise 19. (4 points) For each of the eight DES S-boxes, compute the bias of the random variable $X_2 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4$. (Note that the DES S-boxes map 6-bit input to 4-bit output and are therefore not invertible. But this should not concern you here.)

exe:average-S2

Exercise 20 (the average S-box)**.** For the following S-boxes on $\mathbb{F}_{16}$ draw the linear approximation matrix and find the maximal linear bias:

1. identity id,
2. affine linear transformation $x \mapsto (a^3 + a^2 + 1) \cdot x + (a^2 + a)$,
3. (extended) inverse
$$\mathrm{inv}(x) = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{else,} \end{cases}$$
4. baby-AES S-box.
5. Pick 1 000 S-boxes at random. Draw the distribution of the maximal linear bias.

5. Boolean Functions and Avalanche Criterion

exe:dual **Exercise 21** (The dual of $\mathbb{F}_{2^n}$)**.** Show that every linear boolean function on $\mathbb{F}_{2^n}$ is of the form $x \mapsto \langle a|x\rangle$ for some $a \in \mathbb{F}_{2^n}$. (The vector space of these functions is called the *dual* of $\mathbb{F}_{2^n}$ and denoted $(\mathbb{F}_{2^n})^\vee$).

exe:correlation **Exercise 22** (Correlation)**.** 1. Compute all possible values of $\mathrm{corr}(f, \ell)$ if f and ℓ are linear. [Hint: Without loss of generatility you can assume that f is the zero function.]

2. Use a computer algebra system of your choice to compute the correlations $\mathrm{corr}(\ell_i \circ f_j, \ell_k)$ of the following functions on 8 bits. Hand in a little table for each of the f_j.

   - $f_{-1}(x) = x^{-1}$ for $x \neq 0$ and $f_{-1}(0) = 0$,
   - $f_1(x) = x$,
   - $f_2(x) = x^2$,
   - $f_3(x) = x^3$,
   - $f_*(x) = (a_7 + a_6)a^7 + (a_3 + a_5)a^6 + (a_6 + a_5)a^5 + (a_2 + a_7 + a_4)a^4 + (a_5 + a_7 + a_4 + a_6)a^3 + (a_1 + a_5)a^2 + (a_7 + a_4 + a_6)a + a_6 + a_0 + a_4$ for $x = \sum_{0 \leq i < 8} a_i a^i \in \mathbb{F}_2[a]/\langle a^8 + a^4 + a^3 + a^2 + 1\rangle$.

   - $\ell_0(x) = a_0$,
   - $\ell_1(x) = a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7$,
   - $\ell_2(x) = a_0 + a_4 + a_7$,
   - $\ell_3(x) = a_5 + a_7 + 1$,
   - $\ell_4(x) = a_5 + a_7$

   for $x = \sum_{0 \leq i < 8} a_i a^i \in \mathbb{F}_2[a]/\langle a^8 + a^4 + a^3 + a^2 + 1\rangle$.

exe:avalanche **Exercise 23** (Avalanche)**.** After how many rounds do the following cryptosystems achieve the strict avalanche criterion? [State the properties of the substitution- and permutation-layers that you use and give a complete argument.]

(a) (3 points) baby-AES and (original) AES,

(b) (3 points) a Substitution-Permutation-Network with n-bit input and output.

last: each round n/m S-boxes.

1st round: 1-bit input can effect 2 output bits 2nd round: 2-bit input can effect 4 output bits

$\log_2 n$ rounds to effect each round; +1 for strict avalanche criterion.

Alternatively, $\log_2(n/m)$ such that every input word effects every output word.

exe:average-S3

Exercise 24 (the average S-box)**.** We formulated the following criterion for a "good" S-box:

Flipping a single input bit, flips at least two output bits.

Which of the following S-boxes on $\mathbb{F}_{16}$ satisfies this criterion:

1. identity id,
2. affine linear transformation $x \mapsto (a^3 + a^2 + 1) \cdot x + (a^2 + a)$,
3. (extended) inverse
$$\mathrm{inv}(x) = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{else,} \end{cases}$$
4. baby-AES S-box.
5. Pick 1000 S-boxes at random. What percentage satisfies this criterion?

6. Hash Functions

exe:triplets

Exercise 25. Let $h\colon X \to Y$ be a hash function (assume $\#X >> \#Y$). In lecture we showed that finding a collision for h can be done with $O((\#X)^{1/2})$ random samples of h. How many random samples would it take until we obtain a three way collision, namely distinct strings x,x^*,x' in X such that $h(x) = h(x^*) = h(x')$?

O((\# X)^(2/3))

An informal argument for this is as follows: suppose we collect n random samples. The number of triples among the n samples is n choose 3 which is O(n3). For a particular triple x,y,z to be a 3-way collision we need H(x)=H(y) and H(x)=H(z). Since each one of these two events happens with probability 1/|T| (assuming H behaves like a random function) the probability that a particular triple is a 3-way collision is O(1/|T|2). Using the union bound, the probability that some triple is a 3-way collision is O(n3/|T|2) and since we want this probability to be close to 1, the bound on n follows.

exe:commitment **Exercise 26** (Matching pennies over the phone)**.** The following protocol lets you play "Matching Pennies" over the phone when you have access to a hash function h.

1. Randomly choose a number r
2. Choose your a bit b corresponding to heads/tails and append it to r
3. Compute commitment $x = h(r \mid b)$ and send it to your fellow player.
4. Receive her commitment y
5. Both players reveal their choices and determine the winner.

(a) Assume h is not collision resistant/2nd-preimage-resistant/preimage-resistant. What consequences does it have for the game?

(b) Why is it necessary to prepend your chosen bit b with a random number?

- if not collision resistant, then not binding
- if not preimage resistant, then you always loose

exe:tree-hash **Exercise 27** (Trees as mode of operation)**.** Let $h_0 \colon \{0,1\}^{2m} \to \{0,1\}^m$ be a collision-resistant hash function with $m \in \mathbb{N}_{>0}$.

32.1 (a) (3 points) We construct a hash function $h_1\colon \{0,1\}^{4m} \to \{0,1\}^m$ as follows: Interpret the bit string $x \in \{0,1\}^{4m}$ as $x = (x_1|x_2)$, where both $x_1, x_2 \in \{0,1\}^{2m}$ are words with $2m$ bits. Then compute the hash value $h_1(x)$ as

$$h_1(x) = h_0(h_0(x_1)|h_0(x_2)).$$

Show: h_1 ist collision-resistant.

32.2 (b) (1 point) Let $i \in \mathbb{N}$, $i \geq 1$. We define a hash function $h_i\colon \{0,1\}^{2^{i+1}m} \to \{0,1\}^m$ recursively using h_{i-1} in the following way: Interpret the bit string $x \in \{0,1\}^{2^{i+1}m}$ as $x = (x_1|x_2)$, where both $x_1, x_2 \in \{0,1\}^{2^i m}$ are words with $2^i m$ bits. Then the hash value $h_i(x)$ is defined as

$$h_i(x) = h_0(h_{i-1}(x_1)|h_{i-1}(x_2)).$$

Show: h_i is collision-resistant.

(c) (5 points) The number $p = 2027$ is prime. Now define $h_0 : \{0,1\}^{22} \to \{0,1\}^{11}$ as follows: Let $x = (b_{21}, \ldots, b_0)$ be the binary representation of x. Then $x_1 = \sum_{0 \leq i \leq 10} b_{11+i} 2^i \bmod p$ and $x_2 = \sum_{0 \leq i \leq 10} b_i 2^i \bmod p$. Show that the numbers 5 and 7 have order $p-1$ modulo p. Now compute $y = 5^{x_1} \cdot 7^{x_2} \bmod p$ and let $h_0(x) = (B_{10}, \ldots, B_0)$ be the binary representation of y, i.e. $y = \sum_{0 \leq i < 11} B_i 2^i$. Use the birthday attack to find a collision of h_0 and of h_1 defined as described in (a).

Note: "|" denotes the concatenation of bit strings.

e:variants-of-MD **Exercise 28** (Variants of the Merkle-Damgård construction)**.** Let $n > 1$ be an integer and ℓ a positive polynomial. To obtain a hash function

$$h\colon \bigcup_{0 \leq k \leq \ell(n)} \{0,1\}^k \to \{0,1\}^n,$$

eq:1

we use the Merkle-Damgård construction with a collision-resistant compression function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and a padding scheme as specified below. [Assume initialization vector $IV = 0 \ldots 0$ throughout the exercise.]

(a) (2 points) Messages are *only* padded with 0's until the length is a multiple of the block length. Is h collision-resistant? Prove or provide a counterexample.

(b) (2 points) Messages are padded with a single 1 followed by 0's until the length is a multiple of the block length. This padding includes no information on the message length and the proof of the Merkle-Damgård theorem fails. Let m^* be a block with $f(0, m^*) = 0$. Build collisions for h from this non-collision for f.

(c) (3 points) Messages are padded properly as described in the lecture, *but* the last compression is replaced by a concatenation. In other words, the length of the message is appended to the last intermediate hash, but not hashed itself. Is h collision-resistant? Prove or provide a counterexample.

cor:4

(d) (3 points) Let $n = 4$, $\ell = n^2$, $f\colon \{0,1\}^4 \times \{0,1\}^4 \to \{0,1\}^4, (h_i, m_i) \mapsto$ $h_i \oplus m_i$, and use a padding as described in the lecture. Compute the hash z of $x = 0000$ and provide a 2nd-preimage for x.

(e) (4 points) Find a collision of two equal-length messages for the hash function described in (d). [Hint: This requires at least length 6.]

- 100 and 1000 and 10000
-
-
- $m_0|m_1$ and $m_1|m_0$; $zxor00blocklength$

ash-with-permute

Exercise 29 (Hashing with permutations)**.** Consider a hash function obtained by directly applying the Merkle-Damgård construction (without appending an extra block which encodes the message length) to family of *permutations* π_m. This means that starting from an intermediate hash value h_i and a message block m_i, the next hash value is $h_{i+1} = \pi_{m_i}(h_i)$. The goal of this exercise is to show a weakness of this hash function with respect to the preimage property.

(a) (2 points) Show that when π_m^{-1} is available for every m, preimages can be easily computed if you can choose the initialization vector h_0 at your digression.

This is also true, if h_0 is a fixed value. Consider the following strategy.

item:i
1. Choose a long sequence of message blocks M_i and compute, starting from h_0, the intermediate hash value which we denote by h_i.

item:ii
2. Let h_F be the hash value for which you want to compute a preimage. Choose another long sequence of message blocks M'_i and compute *backwards*, starting from h_F, the intermediate hash values h'_i leading to h_F with the blocks M'_i.

item:iii
3. Stop when h'_i appears in the list of h_i's from step 1.

(b) (2 points) How does the strategy above lead to a preimage for h_F.

(c) (3 points) How would you modify the attack, when proper padding is used?

(d) (5 points) Let π_m be a n-bit block cipher with n-bit keys. What is a reasonable choice for the number of blocks in step 1? How many blocks do you expect to process in step 2 until the condition 3 is satisfied. Compare this to the generic complexity of a brute-force preimage finder.

exe:SHA-bias
Exercise 30 (Bias of the SHA-functions)**.** (4 points) Consider the two non-linear functions MAJ and IF restricted to three bits input (and one bit output). Compute the respective bias.

xe:non-lin-SHA-0
Exercise 31 (Adding non-linearity to the linear model of SHA-0)**.** In the lecture we found 63 non-zero bit sequences of 80 bits that can be used to introduce local collisions in a way that is consistent with the message expansion. Our goal is the maximize the probability of success that such a bit sequence will also yield a collision for the original SHA-0.

We

- choose a bit sequence of small weight, and
- insert it at bit position 1.

1. (2 points) Justify the two criteria. (Remember that bit positions are numbered from 0 to 31.)

Let us track the insertion of a local collision on bit 1 of $W^{(i)}$. For simplicity, assume that the perturbation is $\uparrow$. This affects bit 1 of $A^{(i+1)}$.

If no unwanted carries occur, bit 1 of $A^{(i+1)}$ is also $\uparrow$ and all other remain unchanged. Otherwise, bit 1 of $A^{(i+1)}$ is $\downarrow$ and bit 2 is no longer constant and this may propagate.

2. (1 point) Assume the inputs of the addition are uniformly random values. What is the probability that no carry occurs?

In step $i+2$, the change $\uparrow$ in $A^{(i+1)}$ is invoked (after rotation) in the computation of $A^{(i+2)}$. This is corrected by the change in $W^{(i+2)}$.

3. (1 point) Give a necessary and sufficient condition for the change in bit 6 of $W^{(i+2)}$ such that the correction is performed correctly, i.e. as in the linear model.

In step $i+3$, the change $\uparrow$ is on bit 1 of $B^{(i+2)}$ and involved in the computation of $A^{(i+3)}$. It is corrected with bit 1 of $W^{(i+3)}$. Three cases are possible, after $B^{(i+2)}$ is processed by f (XOR, MAJ, or IF):

- The change $\uparrow$ has vanished,
- the change $\uparrow$ remains unchanged, or
- the change $\uparrow$ has been reversed to a change $\downarrow$

4. (3 points) Compute the probability that a change does not vanish for each of the three possible functions.

The next corrections concern bit 31 of $A^{(i+4)}$ and $A^{(i+5)}$.

5. (1 point) Show that these corrections are always done correctly, if the perturbation does not vanish. In other words, they are done correctly if the change remains unchanged or if the change is reversed.

Finally, the correction on bit 31 of $A^{(i+6)}$ is always correctly canceled by the correction on bit 31 of $W^{(i+5)}$ and we can compute for any round i the probability of successfully applying a single local collision at position 1 in this round.

6. (+4 points) Assume that these probabilities have been computed. Describe a (feasible) strategy to produce collisions from that information.