

COLLEGE OF COMPUTER AND INFORMATION SCIENCE
NORTHEASTERN UNIVERSITY
CSG252: CRYPTOGRAPHY AND COMMUNICATION SECURITY
FALL 2005
PROBLEM SET 2 - SOLUTIONS
PREPARED BY TIM MORGAN

Problem 1 [10 points]:

For each $x, y \in \{1, \dots, n\}$, there exists a unique key $K_{x,y}$ such that $e_{K_{x,y}}(x) = y$. Therefore, $C(K) = \{1, \dots, n\}$ for all $K \in \kappa$. For any $y \in \{1, \dots, n\}$, we have:

$$\begin{aligned} Pr[Y = y] &= \sum_{x \in \{1, \dots, n\}} Pr[\kappa = K_{x,y}] \cdot Pr[X = x] \\ &= \sum_{x \in \{1, \dots, n\}} \left(\frac{1}{n}\right) \cdot Pr[X = x] \end{aligned}$$

$$= \frac{1}{n}$$

Then, for any $x, y \in \{1, \dots, n\}$, we compute:

$$Pr[Y = y | X = x] = Pr[\kappa = K_{x,y}] = \frac{1}{n}$$

Finally, using Bayes' Theorem, we see that:

$$Pr[X = x | Y = y] = Pr[X = x]$$

for all x, y .

Problem 2 [10 points]:

From the given probability distributions, we have:

$$H(K) = 1.585$$

$$H(P) = 1.459$$

Next, we compute the probability distribution on C to be $Pr[(1, 2, 3, 4)] = (\frac{4}{18}, \frac{5}{18}, \frac{6}{18}, \frac{3}{18})$ respectively. Therefore $H(C) = 1.955$. Next we compute:

$$H(K|C) = H(K) + H(P) - H(C) = 1.089$$

In order to compute $H(P|C)$, we first compute $Pr[x|y]$ for all $x \in P$ and all $y \in C$:

	a	b	c
1	$3/4$	0	$1/4$
2	$3/5$	$2/5$	0
3	$1/2$	$1/3$	$1/6$
4	0	$2/3$	$1/3$

From this we compute:

$$\begin{aligned} H(P|1) &= 0.8113 \\ H(P|2) &= 0.9710 \\ H(P|3) &= 1.459 \\ H(P|4) &= 0.9183 \end{aligned}$$

Finally:

$$H(P|C) = \left(\frac{4}{18}, \frac{5}{18}, \frac{6}{18}, \frac{3}{18}\right) \cdot (0.8113, 0.9710, 1.459, 0.9183) = 1.062$$

Problem 3 [10 points]:

$$\begin{aligned} H(K|C) &= \log_2 312 \\ H(K|P, C) &= \log_2 12 \end{aligned}$$

Problem 4 [10 points]:

(a) Using Stirling's approximation, we have:

$$\begin{aligned} \log_2 |\kappa| &\approx \log_2(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n) = n(\log_2 n - \log_2 e) + \frac{1}{2} \cdot \log_2 n + \frac{1}{2} \cdot \log_2(2\pi) \\ &= n(\log_2 n - c_1) + \frac{1}{2} \cdot \log_2 n + c_2 \end{aligned}$$

Where c_1, c_2 are small positive constants. $\log_2 |P| = \log_2 n$, so an estimate for the unicity distance is:

$$\frac{1}{r_L} \left(n - \frac{c_1 n}{\log_2 n} + \frac{1}{2} \right) < \frac{n}{r_L}$$

(b) To simplify things, we will use the estimate $\log_2 n! \approx n \cdot \log_2 n$. Setting $n = 26^m$, we get:

$$\log_2 |\kappa| \approx 26^m \log_2(26^m) = (\log_2 26) \cdot m \cdot 26^m$$

$\log_2 |P| = (\log_2 26) \cdot m$, so the estimate for the unicity distance is:

$$\frac{(\log_2 26) \cdot m \cdot 26^m}{r_L (\log_2 26) \cdot m} \approx 1.33 \cdot 26^m$$

Problem 5 [10 points]:

$$\begin{aligned} & Pr[\kappa_1 + \kappa_2 \bmod 26 = K] \\ &= \sum_{\{(\kappa_1, \kappa_2) \in Z_{26} \times Z_{26} : K \equiv \kappa_1 + \kappa_2 \bmod 26\}} (Pr[\kappa_1 = K_1] \times Pr[\kappa_2 = K_2]) \\ &= \sum_{\{\kappa_2 \in Z_{26}\}} (Pr[\kappa_1 = K - \kappa_2 \bmod 26] \times Pr[\kappa_2 = K_2]) \\ &= \sum_{\{\kappa_2 \in Z_{26}\}} \left(\frac{1}{26} \times Pr[\kappa_2 = K_2]\right) \\ &= \frac{1}{26} \end{aligned}$$

Problem 6 [10 points]:

(This sample solution, implemented in Python, is by Gilberto Molina.)

```
SUBTABLE =
{0: 14, 1:4, 2:13, 3:1,
 4:2, 5:15, 6:11, 7:8,
 8:3, 9:10, 10:6, 11:12,
 12:5, 13:9, 14:0, 15: 7}

PERMTABLE =
{0: 0, 1: 4, 2: 8, 3: 12,
 4: 1, 5: 5, 6: 9, 7: 13,
 8: 2, 9: 6, 10: 10, 11: 14,
 12: 3, 13: 7, 14: 11, 15: 15}

def substitution(num):
    """Given a 16-bit number, return a number
    where every four bits have been substituted
    according to the sub table."""

    hexstring = hex(num)[2:]
    result = ""
    for c in hexstring:
        result += hex(SUBTABLE[int(c, 16)])[2:]

    return int(result,16)

def permutation(num):
    """Given a 16-bit number, return the permutation
    of the bits according to the perm table."""

    result = 0
    for i in range(16):
```

```

    bit_preperm = (num >> (15-i)) & 1
    bit_postperm = bit_preperm << (15-PERMTABLE[i])
    result = result | bit_postperm

return result

def keyschedule(key):
    """Given the original 32-bit key, return a list of
    five 16-bit keys, also known as the key schedule."""

    keys = [ ]
    mask = 0xFFFF
    for i in range(0, 20, 4):
        keys.append((key >> (16-i)) & mask)

    return keys

def encrypt(x, key):
    """Given a 16-bit plaintext and a
    32-bit key, return the 16-bit ciphertext."""

    keys = keyschedule(key)
    w = x
    # do the first three rounds
    for i in range(3):
        u = w ^ keys[i]
        v = substitution(u)
        w = permutation(v)

    # the last round doesn't have permutation
    u = w ^ keys[3]
    v = substitution(u)
    w = v ^ keys[4] # the "whitening" step

    return w

```

Problem 7 [10 bonus points (required for PhD)]:

(a) Since each weighing could have 3 possible outcomes, the most one could learn about the state of the n coins (assuming equal probability of occurrence) is:

$$\sum_{i=1}^3 \left(\frac{1}{3} \cdot \log_2 3\right) = \log_2 3 \approx 1.585$$

bits of information. As each new coin added introduces 2 additional possible states (one each for it being heavier or lighter), the optimal algorithm will be able to produce a correct answer for all situations only if $k \cdot \log_2 3 \geq \log_2(2n)$. One could rewrite this as $n \leq \frac{3^k}{2}$ as an upper bound on n , given a fixed k .

(b) Since there are 12 coins and 2 possible states for the counterfeit coin, there is a total of 24 states in this problem. Therefore, in order to find the answer, we need to learn $\log_2 24 \approx 4.585$ bits of information. Luckily, if we maximize our entropy learned with each weighing, it should be possible to find the answer since $3 \cdot \log_2 3 \approx 4.755$.

By dividing the remaining states up equally each time, we maximize entropy learned and this lets us construct a decision tree of questions which accomplishes this task. (Each possible remaining state is denoted by a coin number, followed by either L or H, signifying the counterfeit being lighter or heavier.)

Initial states possible:

{1L,2L,3L,4L,5L,6L,7L,8L,9L,10L,11L,12L
1H,2H,3H,4H,5H,6H,7H,8H,9H,10H,11H,12H}

First weighing: {1,2,3,4} and {5,6,7,8}

Case (>):

Remaining states: {1H,2H,3H,4H,5L,6L,7L,8L}

Second weighing: {1,2,6} and {3,4,5}

Case (>):

Remaining states: {1H,2H,5L}

Third weighing: {1} and {2}

Case (>): 1 is heavy

Case (<): 2 is heavy

Case (=): 5 is light

Case (<):

Remaining states: {3H,4H,6L}

Third weighing: {3} and {4}

Case (>): 3 is heavy

Case (<): 4 is heavy

Case (=): 6 is light

Case (=):

Remaining states: {7L,8L}

Third weighing: {7} and {8}

Case (>): 8 is light

Case (<): 7 is light

Case (=): impossible

Case (<):

Remaining states: {1L,2L,3L,4L,5H,6H,7H,8H}

Second weighing: {1,2,5} and {3,4,6}

Case (>):

Remaining states: {3L,4L,5H}
 Third weighing: {3} and {4}
 Case (>): 4 is light
 Case (<): 3 is light
 Case (=): 5 is heavy
 Case (<):
 Remaining states: {1L,2L,5H}
 Third weighing: {1} and {2}
 Case (>): 2 is light
 Case (<): 1 is light
 Case (=): 5 is heavy
 Case (=):
 Remaining states: {7H,8H}
 Third weighing: {7} and {8}
 Case (>): 7 is heavy
 Case (<): 8 is heavy
 Case (=): impossible

 Case (=):
 Remaining states: {9L,10L,11L,12L,9H,10H,11H,12H}
 Second weighing: {1,2,3} and {9,10,11}
 Case (>):
 Remaining states: {9L,10L,11L}
 Third weighing: {9} and {10}
 Case (>): 10 is light
 Case (<): 9 is light
 Case (=): 11 is light
 Case (<):
 Remaining states: {9H,10H,11H}
 Third weighing: {9} and {10}
 Case (>): 9 is heavy
 Case (<): 10 is heavy
 Case (=): 11 is heavy
 Case (=):
 Remaining states: {12L,12H}
 Third weighing: {1} and {12}
 Case (>): 12 is light
 Case (<): 12 is heavy
 Case (=): impossible