3.5 (a) Prove that the *Affine Cipher* achieves perfect secrecy if every key is used with equal probability $1/312$.

(b) More generally, suppose we are given a probability distribution on the set

$$\{a \in \mathbb{Z}_{26} : \gcd(a, 26) = 1\}.$$

Suppose that every key (a, b) for the *Affine Cipher* is used with probability $\mathbf{Pr}[a]/26$. Prove that the *Affine Cipher* achieves perfect secrecy when this probability distribution is defined on the keyspace.

Answer:

(a)

3.5(a) Let $P = C = \mathbb{Z}_{26}$ and [Affine cipher].

$K = \{(a,b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} : \gcd(a,26) = 1\}$.

For $k = (a,b) \in K$, define

$$e_k(x) = (ax+b) \bmod 26$$

and $d_k(y) = a^{-1}(y-b) \bmod 26$ for $x, y \in \mathbb{Z}_{26}$

∵ 有 312 keys $= 12 \times 26$, and b can be any number in $\mathbb{Z}_{26}$,

$$Pr[y] = \sum_{k \in K} Pr[k] \cdot Pr[d_k(y)] = 12/312\, Pr[a] + \cdots + 12/312\, Pr[z]$$

$$= 1/26 \cdot (Pr[a] + \cdots Pr[z]) = 1/26$$

By Bayes' Theorem

$$Pr[x|y] = \frac{Pr[x] \cdot Pr[y|x]}{Pr[y]} = \frac{Pr[x] \cdot 1/26}{1/26} = Pr[x]$$

(b)

(b) Still prove from the definition of Affine cipher

即 $p(y|x) = p(y)$ 证明独立性.

∵ $e_k(x) = (ax+b) \bmod 26$

$Pr(b) = \frac{1}{26}$ ($b\ [0, 26] \in \mathbb{Z}$)

每个 a $\frac{Pr(a)}{26}$ 又∵ $\sum_{b \in \mathbb{Z}_{26}} p(b) = 1$

$$p(y) = \sum_{a \in \mathbb{Z}_{26}} \sum_{b \in \mathbb{Z}_{26}} p(x, a, b)$$

$$= \sum_{a \in \mathbb{Z}_{26}} \sum_{b \in \mathbb{Z}_{26}} p(x) \cdot p(a) \cdot p(b)$$

$$= \sum_{a \in \mathbb{Z}_{26}} Pr[a] \cdot \sum_{b \in \mathbb{Z}_{26}} \frac{p(x)}{26^2}$$

又∵ $p(b) = \frac{1}{26}$ $\sum_{b \in \mathbb{Z}_{26}} p(b) = 1$

$$\therefore p(y) = \sum_{a \in \mathbb{Z}_{26}^*} \frac{Pr[a]}{26} \cdot p(x)$$

$Pr[a]$ is known, y depend on $p(x)$ not itself ∴ perfect secrecy

3.8 Suppose that y and y' are two ciphertext elements (i.e., binary n-tuples) in the *One-time Pad* that were obtained by encrypting plaintext elements x and x', respectively, using the same key, K. Prove that $x + x' \equiv y + y' \pmod 2$.

Answer:

3.8 Prove $x+x' \equiv y+y' \ (\mathrm{mod}\ 2)$ One-time Pad

∵明文 x，密钥 k，密文 y $\quad y = x \oplus k$

∴ $y = x \oplus k$

$y' = x' \oplus k$

∴ $y \oplus y' = (x \oplus k) \oplus (x' \oplus k)$

By associative law

∴ $= (x \oplus x') \oplus (k \oplus k') = 0$

∴ $= x \oplus x' \oplus 0$

$= x \oplus x'$

Conclusion $y \oplus y' = x \oplus x'$

3.9 (a) Construct the encryption matrix (as defined in Example 3.3) for the *One-time Pad* with $n = 3$.

```
import numpy as np
# Size of the binary tuple
n = 3
# 生成所有 n=3 可能的二元组
binary_tuples = np.array([[int(x) for x in format(i, f'0{n}b')] for i in
range(2**n)])
# 初始化大小为 2^n x 2^n 的空加密矩阵
encryption_matrix = np.zeros((2**n, 2**n), dtype=int)
# 构造加密矩阵
# 加密矩阵中的每一项都是表示明文和密钥的二元元组的异或(XOR)
encryption_matrix_output = ""
for i, plaintext in enumerate(binary_tuples):
for j, key in enumerate(binary_tuples):
encryption_matrix[i, j] = int(''.join(str((p ^ k)) for p, k in zip(plaintext,
key)), 2)
# 将矩阵行添加到输出字符串中
encryption_matrix_output += " ".join(str(num) for num in
encryption_matrix[i, :]) + "\n"
```

```
print(encryption_matrix_output)
encryption_matrix
```

Output:

```
● (base) wangyidan@wangyidandeMacBook-Pro hw2 % /usr/local/bin/python3 /Users/wangyidan/Desktop/密码学/hw2/3.9.py
0 1 2 3 4 5 6 7
1 0 3 2 5 4 7 6
2 3 0 1 6 7 4 5
3 2 1 0 7 6 5 4
4 5 6 7 0 1 2 3
5 4 7 6 1 0 3 2
6 7 4 5 2 3 0 1
7 6 5 4 3 2 1 0
```

3.15 Consider a cryptosystem in which $\mathcal{P} = \{a, b, c\}$, $\mathcal{K} = \{K_1, K_2, K_3\}$ and $\mathcal{C} = \{1, 2, 3, 4\}$. Suppose the encryption matrix is as follows:

	a	b	c
K_1	1	2	3
K_2	2	3	4
K_3	3	4	1

Given that keys are chosen equiprobably, and the plaintext probability distribution is $\mathbf{Pr}[a] = 1/2$, $\mathbf{Pr}[b] = 1/3$, $\mathbf{Pr}[c] = 1/6$, compute $H(\mathbf{P})$, $H(\mathbf{C})$, $H(\mathbf{K})$, $H(\mathbf{K}|\mathbf{C})$, and $H(\mathbf{P}|\mathbf{C})$.

Answer:

3.15. 明文集合 $P = \{a,b,c\}$，密钥集合 $K = \{k_1,k_2,k_3\}$ 各 $\frac{1}{3}$（密钥出现等概率），密文集合 $C = \{1,2,3,4\}$

又∵ 信息熵 $H(X) = -\sum_i Pr[x_i]\cdot\log_2 Pr[x_i]$

$Pr[a] = \frac{1}{2}$ $Pr[b] = \frac{1}{3}$ $Pr[c] = \frac{1}{6}$

	a	b	c
k_1	1	2	3
k_2	2	3	4
k_3	3	4	1

$H(P)$ 明文. $H(K)$ 密钥 $H(C)$ 密文.

$H(K|C)$ 已知密文，密钥的概率

$H(P) = -\frac{1}{2}\cdot\log_2\frac{1}{2} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{6} = 1.4591$

$H(k) = \left[-\frac{1}{3}\log_2\frac{1}{3}\right]\times 3 = 1.5850$

$Pr[1] = Pr[k_1]\times Pr[a] + Pr[k_3]\times Pr[c] = \frac{1}{2}\times\frac{1}{3} + \frac{1}{6}\times\frac{1}{3} = \frac{2}{9}$

$Pr[2] = Pr[k_1]\times Pr[b] + Pr[k_2]\times Pr[a] = \frac{1}{3}\times\frac{1}{3} + \frac{1}{3}\times\frac{1}{2} = \frac{5}{18}$

$Pr[3] = Pr[k_1]\times Pr[c] + Pr[k_2]\times Pr[b] + Pr[k_3]\times Pr[a] = \frac{1}{3}\times\frac{1}{6} + \frac{1}{3}\times\frac{1}{3} + \frac{1}{3}\times\frac{1}{2} = \frac{1}{3}$

$Pr[4] = Pr[k_2]\times Pr[c] + Pr[k_3]\times Pr[b] = \frac{1}{3}\times\frac{1}{6} + \frac{1}{3}\times\frac{1}{3} = \frac{1}{6}$

$H(C) = -\frac{2}{9}\cdot\log_2\frac{2}{9} - \frac{5}{18}\cdot\log_2\frac{5}{18} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{6}\log_2\frac{1}{6} = 1.9547.$

$H(K|C) = 1.0894$ (根据python)

Python:

```
import numpy as np

# 明文
prob_P = {'a': 1/2, 'b': 1/3, 'c': 1/6}
# 每个密钥的概率相等各位 1/3
```

```
prob_K = 1/3
# 明文的熵
H_P = -sum(prob * np.log2(prob) for prob in prob_P.values())
# 密钥熵(均匀分布熵的公式)
H_K = -3 * (prob_K * np.log2(prob_K))
# 输出结果
print(H_P, H_K)
# 给定加密矩阵作为一个字典
encryption_matrix = {
'K1': {'a': 1, 'b': 2, 'c': 3},
'K2': {'a': 2, 'b': 3, 'c': 4},
'K3': {'a': 3, 'b': 4, 'c': 1}
}
# 计算 C 的概率分布
prob_C = {1: 0, 2: 0, 3: 0, 4: 0}
for plaintext, p in prob_P.items():
for key, encrypted_values in encryption_matrix.items():
prob_C[encrypted_values[plaintext]] += p * prob_K
# 密文的熵
H_C = -sum(prob * np.log2(prob) for prob in prob_C.values())
# 用 K 和 C 的联合概率分布，从 encryption_matrix 中推导出来
# 计算条件熵 H(K|C)
# 初始化联合概率字典
joint_prob_KC = {k: {1: 0, 2: 0, 3: 0, 4: 0} for k in encryption_matrix}
# 计算每个 K 和 C 的联合概率
for key, mappings in encryption_matrix.items():
for plaintext, encrypted_value in mappings.items():
joint_prob_KC[key][encrypted_value] += prob_P[plaintext] * prob_K
# 计算 C 的边际概率
marginal_prob_C = {c: sum(joint_prob_KC[key][c] for key in
joint_prob_KC) for c in prob_C}
# 计算条件熵 H(K|C)
H_K_given_C = 0
for key in joint_prob_KC:
for c in joint_prob_KC[key]:
if joint_prob_KC[key][c] > 0:
H_K_given_C -= joint_prob_KC[key][c] * np.log2(joint_prob_KC[key][c]
/ marginal_prob_C[c])
# 输出结果
print(H_C, H_K_given_C)
```

Output:

```
(base) wangyidan@wangyidandeMacBook-Pro hw2 % /usr/local/bin/python3 "/Users/wangyidan/Desktop/密码学/hw2/3.15(2).py"
1.4591479170272448 1.584962500721156
1.9546859469463558 1.089424470802045
```

3.17 Suppose that *APNDJI* or *XYGROBO* are ciphertexts that are obtained from encryption using the *Shift Cipher*. Show in each case that there are two "meaningful" plaintexts that could encrypt to the given ciphertext. (Thanks to John van Rees for these examples.)

```
# 定义一个函数，使用给定的密钥使用移位密码解密密文
def decrypt_shift_cipher(ciphertext, key):
decrypted_text = ""
for char in ciphertext:
if char.isalpha(): # 检查字符是否为字母表
shift = (ord(char) - key - 65) % 26 + 65 # 按键值向后移
decrypted_text += chr(shift)
else:
decrypted_text += char
return decrypted_text
# 给定暗文
ciphertexts = ['APNDJI', 'XYGROBO']
# 因为我们不知道 shift/key 的值，所以我们尝试从 1 到 25 的所有可能性
# 对于一个有效的移位密码，密钥永远不会是 0，因为那意味着没有加密，也不会是 26，因为它是一个完整的循环。
for ciphertext in ciphertexts:
print(f"Trying decryption for {ciphertext}:")
for key in range(1, 26):
decrypted = decrypt_shift_cipher(ciphertext, key)
print(f"Key {key}: {decrypted}")
print("\n")
```

Output:

```
(base) wangyidan@wangyidandeMacBook-Pro hw2 % /usr/local/bin/python3 /Users/wangyidan/Desktop/密码学/hw2/3.17.py
Trying decryption for APNDJI:
Key 1: ZOMCIH
Key 2: YNLBHG
Key 3: XMKAGF
Key 4: WLJZFE
Key 5: VKIYED
Key 6: UJHXDC
Key 7: TIGWCB
Key 8: SHFVBA
Key 9: RGEUAZ
Key 10: QFDTZY
Key 11: PECSYX
Key 12: ODBRXW
Key 13: NCAQWV
Key 14: MBZPVU
Key 15: LAYOUT
Key 16: KZXNTS
Key 17: JYWMSR
Key 18: IXVLRQ
Key 19: HWUKQP
Key 20: GVTJPO
Key 21: FUSION
Key 22: ETRHNM
Key 23: DSQGML
Key 24: CRPFLK
```

```
Key 25: BQOEKJ


Trying decryption for XYGROBO:
Key 1: WXFQNAN
Key 2: VWEPMZM
Key 3: UVDOLYL
Key 4: TUCNKXK
Key 5: STBMJWJ
Key 6: RSALIVI
Key 7: QRZKHUH
Key 8: PQYJGTG
Key 9: OPXIFSF
Key 10: NOWHERE
```

```
Key 11: MNVGDQD
Key 12: LMUFCPC
Key 13: KLTEBOB
Key 14: JKSDANA
Key 15: IJRCZMZ
Key 16: HIQBYLY
Key 17: GHPAXKX
Key 18: FGOZWJW
Key 19: EFNYVIV
Key 20: DEMXUHU
Key 21: CDLWTGT
Key 22: BCKVSFS
Key 23: ABJURER
Key 24: ZAITQDQ
Key 25: YZHSPCP
```