

Überschrift

Oliver Oberdick

5. Oktober 2025

Zusammenfassung

Beschreibung zur APL Programmierung 2 der Gruppe HH05

Inhaltsverzeichnis

1	Story	1
1.1	Levelvorbereitung	1
2	Levelbeschreibung	2
2.1	Level 1	2
2.2	Level 2	2
2.3	Level 3	2
2.4	Level 4	2
2.5	Level 5	2
2.6	Level 6	2
A	Programmcode	3
A.1	Vorbereitungen	3
A.2	Beispiellösungen	8

1 Story

Du schaltest deinen Computer ein und siehst nur eine eigenartige Eingabemaske vor dir. Irgendwer Irgendetwas scheint ihn verändert zu haben. Jetzt mußt du die Kontrolle zurückerlangen. 'Freundlicherweise' wurden dir einige Hinweise hinterlassen. Finde und nutze sie!

1.1 Levelvorbereitung

1: erstellen eines zufälligen Verschlüsselungs Key in einer Variable oder speichern in einer Datei 'Dateiname Sec_Key_HH05.key' zur weiteren Nutzung.

2: evtl könnte auf die Speicherung des Key in einer Datei verzichtet werden, wenn die Eingaben für die Level 1-?? vor dem Level Start generiert werden, dann könnte der Key in einer Variable verbleiben

3: Verstecken des Key in einer Zufällig ausgewählten Bild-Datei (Eingabedaten für Level 2) Bild-Datei immer wieder neu erstellen und die alte überschreiben.

4: generieren einer Log-Datei Bsp. wie die Ausgabe von
netstat -l — grep LISTEN
mit zufällig ergänzten Portnummern, welche nicht offen sein sollten.

5: Nutzen des Key für die Verschlüsselung der Log-Datei auf Bitebene (Eingabedaten für Level 4)

6: ?

2 Levelbeschreibung

2.1 Level 1

Bsp.: Du schaltest dein Rechner ein und bekommst nur einen eigenartigen Eingabeprompt

- ??

2.2 Level 2

‘Finde das flag / den Key in dem Bild

- auf einer normalen Kommandozeile könntest du ihn mit dem Kommando Strings auslesen.
- merke dir den Key gut.
- ?
- nur in dieser Runde lautet der Key ???

2.3 Level 3

Bsp.: Deine Dateien sind nicht da wo sie sein sollten, suche sie!

- ??

2.4 Level 4

die gefundene Datei muß entschlüsselt werden.

- Vielleicht könnte dir der “gemerkte Key” dir behilflich sein.
- Verschlüsselung mit XOR
- die Verschlüsselung erfolgte auf Bitebene.
- wenn der Key nicht lang genug ist, Hilft vielleicht eine aneinanderreihung

2.5 Level 5

Bsp: Mal schauen was in der Datei zu finden ist

- ??

2.6 Level 6

Bsp: Was sollte jetzt mit den gefundenen Informationen angestellt werden.

- ??

A Programmcode

A.1 Vorbereitungen

Generell:

```
1 import random
2 import string
3 from EscapeRoom import EscapeRoom
4
5 import lib.stego as STEGO # Funktionssammlung Oliver Level 2
6 import lib.crypt as CRYPT # Funktionssammlung Oliver Level 4
7
8 class Gruppe_HH_05(EscapeRoom):
9
10     def __init__(self):
11         super().__init__()
12         self.set_metadata("Veronika, Lucasz & Oliver", __name__)
13         self.key = CRYPT.schlüssel_erstellen(30) #schlüssel erstellen
14         self.bild = "static/KEY.jpg"
15         STEGO.random_bild(self.bild) # zufaelliges Bild ermitteln und umkopieren
16         STEGO.im_bild_verstecken(self.bild, self.key)
17         self.verschlusselt = "static/text.crypt"
18         CRYPT.schlüsselanwendung_datei("static/originale/test.log", self.verschlusselt,
19 self.key)
20
21         self.add_level(self.create_level1()) # Veronika
22         self.add_level(self.create_level2()) # Oliver
23         self.add_level(self.create_level3()) # Veronika
24         self.add_level(self.create_level4()) # Oliver
25         self.add_level(self.create_level5()) # Lucasz
26         self.add_level(self.create_level6()) # Lucasz
27
28     ### LEVELS ###
29     # Level 1
30     def create_level1(self):
31         task_messages = [
32             " ",
33             "Hi,",
34             "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen",
35         ]
36         hints = [
37             "schau mal im Bild!",
38             "suche nach dem flag= ",
39             "Eingabedaten sind der Dateiname des Bildes",
40             "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
41             "speichern kann nicht schaden, Vorschlag game.key",
42             "als encoding wurde 'ISO-8859-1' verwendet",
43             "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
44         ]
45         return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
46 im_bild_finden, "data": self.bild}
47
48     # Level 2
49     def create_level2(self):
50         task_messages = [
51             " ",
52             "Hi,",
53             "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen"
54         ]
55         hints = [
56             "schau mal im Bild!",
57             "suche nach dem flag= ",
58             "Eingabedaten: Dateiname des Bildes",
```

```

57         "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
58         "speichern kann nicht schaden, Bsp. game.key",
59         "als encoding wurde 'ISO-8859-1' verwendet",
60         "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
61     ]
62     return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
im_bild_finden, "data": self.bild}
63
64 # Level 3
65 def create_level3(self):
66     task_messages = [
67         " ",
68         "Hi,",
69         "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen",
70     ]
71     hints = [
72         "schau mal im Bild!",
73         "suche nach dem flag= ",
74         "Eingabedaten sind der Dateiname des Bildes",
75         "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
76         "speichern kann nicht schaden, Bsp. game.key",
77         "als encoding wurde 'ISO-8859-1' verwendet",
78         "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
79     ]
80     return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
im_bild_finden, "data": self.bild}
81
82 # Level 4
83 def create_level4(self):
84     task_messages = [
85         "Du hast jetzt einen Dateinamen " + self.verschlusselt + ", schon mar
reingeschaut?",
86         "zur kontrolle, zeig mir die Zeichen 20 - 70"
87     ]
88     hints = [
89         "kannst du den Inhalt lesen?",
90         "Hattest du die flag gespeichert? Bsp. game.key?",
91         "Bitweises XOR schon mal gesehen?",
92         "als Rueckgabewert die Zeichen 20 - 70 als String zum alsolvieren dieses Level
sollten erstmal reichen",
93         "Denke drann den Inhalt des Key.File zu nutzen, nicht den Dateinamen",
94         "den Key kannst du auch mehrfach hintereinander schreiben, falls er nicht lang
genug ist",
95         "trotzdem solltest du die komplette Datei bearbeiten und auch wieder speichern.
Bsp. ausgabe_encrypt.txt"
96     ]
97     return {"task_messages": task_messages, "hints": hints, "solution_function": CRYPT.
entschluesseln, "data": self.verschlusselt}
98
99 # Level 5
100 def create_level5(self):
101     task_messages = [
102         " ",
103         "Hi,",
104         "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen"
105     ]
106     hints = [
107         "schau mal im Bild!",
108         "suche nach dem flag= ",
109         "Eingabedaten sind der Dateiname des Bildes",
110         "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
111         "speichern kann nicht schaden, Vorschlag game.key",
112         "als encoding wurde 'ISO-8859-1' verwendet",
113         "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
114     ]

```

```

115         return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
116             im_bild_finden, "data": self.bild}
117
118     # Level 6
119     def create_level6(self):
120         task_messages = [
121             "  ",
122             "Hi,",
123             "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen"
124         ]
125         hints = [
126             "schau mal im Bild!",
127             "suche nach dem flag= ",
128             "Eingabedaten sind der Dateiname des Bildes",
129             "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
130             "speichern kann nicht schaden, Vorschlag game.key",
131             "als encoding wurde 'ISO-8859-1' verwendet",
132             "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
133         ]
134         return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
135             im_bild_finden, "data": self.bild}
136
137     ### Hilfsfunktionen ###
138
139     ### SOLUTIONS ###

```

Listing 1: Genereller Escape-Room

Level 1

Level 2

```

1  #!/usr/bin/python3
2
3  import random
4  import os
5
6  # """ Steganographie
7  # verstecken und auslesen von Nachrichten in einem Bild.
8  #
9  # Oliver Oberdick Matrikel:548933
10 # """
11
12 # Hilfsfunktion nur fuer den EscapeRoom, damit unterschiedliche Bilder genutzt werden
13 def random_bild(ziel_bild):
14     nummer = random.randint(1, 9)
15     dst = ziel_bild
16     src = "static/original/Bild_Schluessel_" + str(nummer) + ".jpg"
17     if os.name == 'nt': # pruefen ob Windows
18         kopierbefehl = f'copy "{src}" "{dst}"'
19     else:
20         kopierbefehl = f'cp "{src}" "{dst}"'
21     os.system(kopierbefehl)
22
23 # Funktion zum Vorbereiten der Level
24 def im_bild_verstecken(bild_datei, schluessel):
25     bild = open(bild_datei, encoding="ISO-8859-1", mode="a+")
26     bild.write("flag=" + schluessel)
27     bild.close()
28
29 # Kontrollfunktion

```

```

30 def im_bild_finden(bild_datei, was="flag="):
31     bild = open(bild_datei, encoding="ISO-8859-1", mode="r")
32     search = was
33     try:
34         txt = ""
35         byte = bild.read(1)
36         while byte != "":
37             txt = txt + byte
38             byte = bild.read(1)
39         pos = txt.find(search) # position des suchstring finden
40         pos = pos + len(search) # laenge des suchstrings ueberspringen
41         with open("tmp/game.key", 'w') as tmp: # gefundenen Schluessl zwischenspeichern
42             tmp.writelines(txt[pos:])
43         bild.close()
44         return txt[pos:]
45     except:
46         bild.close()
47
48 ##
49 if __name__ == "__main__":
50     pass

```

Listing 2: Funktionalitäten für Level 2

Level 3

Level 4

```

1  #!/usr/bin/python3
2
3  import random
4
5  # """ Verschluesselung
6  # Symmetrische Verschluesselung mittels XOR auf Bit-Basis
7  #
8  # Oliver Oberdick Matrikel:548933
9  # """
10
11 # Hilfsfunktion zum erstellen eines Verschluesselungs key in beliebiger laenge
12 def schluessel_erstellen(laenge):
13     ergebnis = ""
14     while len(ergebnis) < laenge:
15         zahl = random.randint(48, 122)
16         if ((zahl >= 48 and zahl <= 57) or (zahl >= 65 and zahl <= 90) or (zahl >= 97
17         and zahl <= 122)):
18             # Damit der Schluessel nur aus Zahlen, Grossbuchstaben und Kleinbuchstaben besteht
19             ergebnis += chr(zahl)
20         return ergebnis
21
22 def string_to_binaer(nachricht):
23     ergebnis = ""
24     for c in nachricht:
25         ergebnis += ''.join(format(ord(c), '08b'))
26     return ergebnis
27
28 def binaer_to_string(nachricht):
29     ergebnis = ""
30     for i in range(0, len(nachricht), 8):
31         ergebnis += chr(int(nachricht[i: i+8], 2))
32     return ergebnis
33
34 # Ver oder Entschluesseln eines String mittels XOR (Symmetrisch)
35 def schluesselanwendung(was, womit):

```

```

35     ergebnis = ""
36     schluessel = ""
37     while len(schluessel) < len(was):
38         schluessel += womit
39     binaer_schluessel = string_to_binaer(schluessel)
40     binaer_nachricht = string_to_binaer(was)
41     for i in range(len(binaer_nachricht)):
42         ergebnis += str(int(binaer_nachricht[i]) ^ int(binaer_schluessel[i]))
43     return binaer_to_string(ergebnis)
44
45 # erweiterung, damit auch Dateien ver und entschluesselt werden koennen
46 def schluesselanwendung_datei(eingabe_datei, ausgabe_datei, schluessel):
47     counter = 0 # nur zur kontrolle
48     ergebnis = "" # nur zur kontrolle
49     with open(eingabe_datei, 'r') as in_file:
50         with open(ausgabe_datei, 'w') as out_file:
51             for line in in_file.read():
52                 counter += 1 # nur zur kontrolle
53                 tmp = schluesselanwendung(line, schluessel)
54                 out_file.write(tmp)
55                 if (counter >= 20 and counter >= 70): # nur zur kontrolle
56                     ergebnis += tmp # nur zur kontrolle
57     return ergebnis # nur zur kontrolle im EscapeRoom Spiel
58
59 # Angepasste Funktion, damit zum entschluesseln der Schluessel aus einer Daten genutzt
60   werden kann
61 def entschluesseln(eingabe, ausgabe="tmp/ausgabe_encrypt.txt", schluessel="tmp/game.key"):
62     key = ""
63     with open(schluessel, "r") as f:
64         key = f.readline()
65     return schluesselanwendung_datei(eingabe, ausgabe, key)
66
67 ##
68 if __name__ == "__main__":
69     key1 = schluessel_erstellen(20)
70     key2 = schluessel_erstellen(20)
71
72     print(key1)
73     print(key2)
74
75     text = "Hallo du da im Radio!"
76
77     print("Original Text")
78     print(text)
79     print("Verschluesselt mit Key1")
80     text_verschluesselt = schluesselanwendung(text, key1)
81     print(text_verschluesselt)
82     print("Entschluesselt mit Key1")
83     text_entschluesselt = schluesselanwendung(text_verschluesselt, key1)
84     print(text_entschluesselt)
85     print("entschluesselt mit Key2")
86     text_entschluesselt = schluesselanwendung(text_verschluesselt, key2)
87     print(text_entschluesselt + " - mit falschem Key")
88
89     print("-----")
90
91     schluesselanwendung_datei("test.txt", "test.crypt", key1)
92
93     schluesselanwendung_datei("test.crypt", "test_decrypt.txt", key1)
94
95     print("Dateien fertig")

```

Listing 3: Funktionalitäten für Level 4

Level 5

Level 6

A.2 Beispiellösungen

Level 1

Level 2

```
1 # Beispielloesung Level 2
2
3 def run(wo, was="flag="):
4     bild = open(wo, encoding="ISO-8859-1", mode="r")
5     search = was
6     try:
7         txt = ""
8         byte = bild.read(1)
9         while byte != "":
10             txt = txt + byte
11             byte = bild.read(1)
12         pos = txt.find(search)
13         pos = pos + len(search)
14         with open("tmp.txt", 'w') as tmp: # gefundenen Schluessel zwischenspeichern
15             tmp.writelines(txt[pos:])
16         return txt[pos:]
17     except:
18         bild.close()
```

Listing 4: Beispiellösung Aufgabe 2

Level 3

Level 4

```
1 # Beispielloesung Level 4
2
3 def run(eingabe):
4     return schluesselanwendung_datei(eingabe, "ausgabe_encrypt.txt", "tmp.txt")
5
6 def string_to_binaer(nachricht):
7     ergebnis = ""
8     for c in nachricht:
9         ergebnis += ''.join(format(ord(c), '08b'))
10    return ergebnis
11
12 def binaer_to_string(nachricht):
13    ergebnis = ""
14    for i in range(0, len(nachricht), 8):
15        ergebnis += chr(int(nachricht[i: i+8], 2))
16    return ergebnis
17
18 def schluesselanwendung(was, womit):
19    ergebnis = ""
20    schluessel = ""
21    while len(schluessel) < len(was):
22        schluessel += womit
23    binaer_schluessel = string_to_binaer(schluessel)
24    binaer_nachricht = string_to_binaer(was)
25    for i in range(len(binaer_nachricht)):
26        ergebnis += str(int(binaer_nachricht[i]) ^ int(binaer_schluessel[i]))
```

```

27     return binaer_to_string(ergebnis)
28
29 def schluesselanwendung_datei(eingabe_datei, ausgabe_datei, schluessel):
30     key = ""
31     with open(schluessel, "r") as f:
32         key = f.readline()
33     counter = 0
34     ergebnis = ""
35     with open(eingabe_datei, 'r') as in_file:
36         with open(ausgabe_datei, 'w') as out_file:
37             for line in in_file.read():
38                 counter += 1
39                 tmp = schluesselanwendung(line, key)
40                 out_file.write(tmp)
41                 if (counter >= 20 and counter >= 70):
42                     ergebnis += tmp
43     return ergebnis

```

Listing 5: Beispiellösung Aufgabe 4

Level 5

Level 6