Exercises I

Computer and Network Security

Emilio Coppa

Disclaimer. The following exercises are taken from the internet or other public sources (e.g., books). At the exam, you may get exercises that are similar, hence it makes sense to actually solve them by yourself (or at least try!). At the exam your answer should discuss the solution step by step.

Exercise 1.1 Understanding Cryptography

Decrypt the following text that has been encrypted with a substitution cipher:

lrvmnir bpr sumvbwvr jx bpr lmiwv yjeryrkbi jx qmbm wi bpr xjvni mkd ymibrut jx irhx wi bpr riirkvr jx ymbinlmtmipw utn qmumbr dj w ipmhh but bj rhnvwdmbr bpr yjeryrkbi jx bpr qmbm mvvjudwko bj yt wkbrusurbmbwjk lmird jk xjubt trmui jx ibndt wb wi kjb mk rmit bmiq bj rashmwk rmvp yjeryrkb mkd wbi iwokwxwvmkvr mkd ijyr ynib urymwk nkrashmwkrd bj ower m vjyshrbr rashmkmbwjk jkr cjnhd pmer bj lr fnmhwxwrd mkd wkiswurd bj invp mk rabrkb bpmb pr vjnhd urmvp bpr ibmbr jx rkhwopbrkrd ywkd vmsmlhr jx urvjokwgwko ijnkdhrii ijnkd mkd ipmsrhrii ipmsr w dj kjb drry ytirhx bpr xwkmh mnbpjuwbt lnb yt rasruwrkvr cwbp qmbm pmi hrxb kj djnlb bpmb bpr xjhhjcwko wi bpr sujsru msshwvmbwjk mkd wkbrusurbmbwjk w jxxru yt bprjuwri wk bpr pjsr bpmb bpr riirkvr jx jqwkmcmk qmumbr cwhh urymwk wkbmvb

Exercise 1.1 Understanding Cryptography: solution

Because the practice of the basic movements of kata is the focus and mastery of self is the essence of matsubayashi ryu karate do i shall try to elucidate the movements of the kata according to my interpretation based on forty years of study. It is not an easy task to explain each movement and its significance and some must remain unexplained to give a complete explanation one would have to be qualified and inspired to such an extent that he could reach the state of enlightened mind capable of recognizing soundless sound and shapeless shape i do not deem myself the final authority but my experience with kata has left no doubt that the following is the proper application and interpretation i offer my theories in the hope that the essence of okinawan karate will remain intact.

Exercise Alphabet Soup

Credits: NeverlanCTF-2019

Description: MKXU IDKMI DM BDASKMI NLU XCPJNDICFQ! K VDMGUC KW PDT GKG NLKB HP LFMG DC TBUG PDTC CUBDTCXUB. K'Q BTCU MDV PDT VFMN F WAFI BD LUCU KN KB WAFI GDKMINLKBHPLFMGKBQDCUWTMNLFMFMDMAKMUNDDA

Exercise Alphabet Soup: hint

Idea: perform frequency analysis of the string

Exercise Alphabet Soup: solution

The string is encrypted using a substitution cipher. The plaintext is:

NICE GOING ON SOLVING THE CRYPTOGRAM! I WONDER IF YOU DID THIS BY HAND OR USED YOUR RESOURCES. I'M SURE NOW YOU WANT A FLAG SO HERE IT IS FLAG DOINGTHISBYHANDISMOREFUNTHANANONLINETOOL

Credits: str0nkus

Exercise readyXORnot

Credits: TamuCTF

Description:

original data: "El Psy Congroo"
encrypted data: "IFhiPhZNYiOKWiUcCls="
encrypted flag: "I3gDKVh1Lh4EVyMDBFo="

The flag is not in the traditional gigem{flag} format.

Exercise readyXORnot: solution

If we have two ciphertext xor'ed with the same key is very easy to get the key. The only detail is that the two ciphertexts are in base64. In python (credits):

```
import base64
from itertools import izip, cycle
original = "El Psy Congroo"
enc_data = base64.decodestring("IFhiPhZNYiOKWiUcCls=")
enc_flag = base64.decodestring("I3gDKVh1Lh4EVyMDBFo=")

def xor(data, key):
        return ''.join(chr(ord(x) ^ ord(y)) for (x,y) in izip(data, cycle(key)))

key = xor(enc_data, original)
print(xor(enc_flag, key))
```

Exercise Hidden Flag

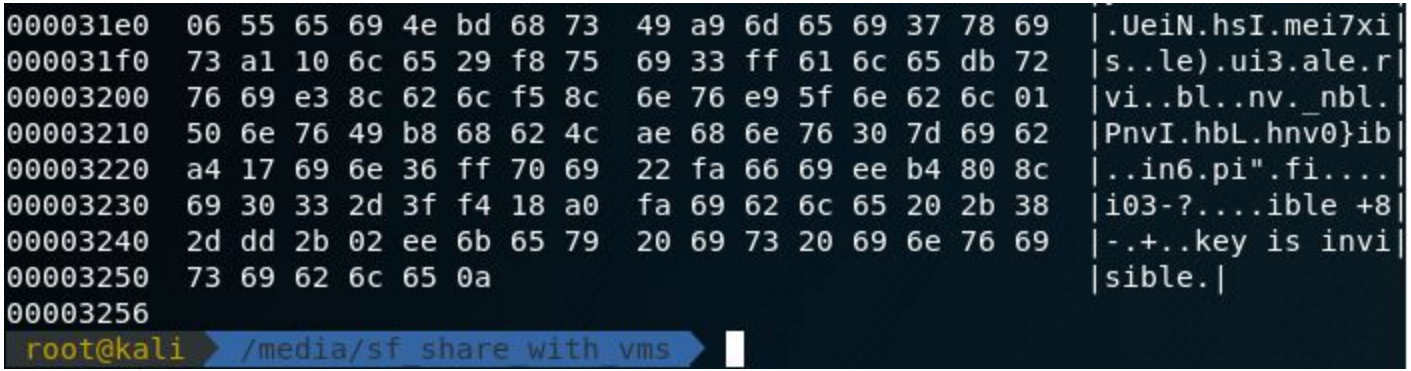
Credits: CTF HSCTF 6

Input: an image chall.png

Goal: find the hidden flag

Exercise Hidden Flag: hint #1

Idea: if we look at the hex representation of the file we get an hint



“key is invisible” seems to suggest the value of the key. However, which cipher?

Exercise Hidden Flag: solution

Using the xortool we can easily test different keys:

```
root@kali /media/sf_share_with_vms xortool chall.png -l 9 -b
256 possible key(s) of length 9:
invisible
howhrhcmd
kltkqk`ng
jmujpjaof
mjrmwmfha
...
Found 0 plaintexts with 95.0%+ valid characters
See files filename-key.csv, filename-char_used-perc_valid.csv
```

when checking the file xortool_out/000.out (created by xortool) we see an image with the flag!

Credits: kurawa2

Exercise A Lost Cause

Credits: CTF HSCTF 6

Description: Pirate Keith loves cryptography and has protected his treasure with a very annoying caesar shift. He has witten

“CGULKVIPFRGDOOCSJTRRVMORCQDZG”

on his treasure chest and has left a piece of paper with the following message: “every subsequent letter is shifted one less than the previous.” Knowing this, can you unlock Pirate Keith’s treasure chest?

Exercise A Lost Cause: solution

This is a simple Caesar's cipher (credits: kurawa2):

```
enc='CGULKVIPFRGDOOCSJTRRVMORCQDZG'
for i in range(26):
   ans=''
   for j in range(len(enc)):
       d = ord(enc[j]) - i - (26-j)
       while d < 65:
           d += 26
       ans += chr(d)
   print(ans)
```

which prints for a specific j value the string "GLASSESAREUSEFULDONOTLOSETHEM"

Exercise streamgame1

Credits: QWB 2018

Description: Decrypt the following ciphertext:

"55 38 F7 42 C1 0D B2 C7 ED E0 24 3A"

which was generated using the following Python [code].

Can you recover the key (flag variable in the code)?

Exercise streamgame1

The code is somehow implementing a simple Linear Shift Feedback Register. Solutions:

- two different approaches: see here (translate the page into english!)
- bruteforce solutions on Piazza CNS by students:
  - Matteo Marino
  - Francesco Bovi

Exercise MTV Cribs

Credits: RickAndMortyCTF 2018

Description: Two messages have been encrypted with OTP using the same key:

"382d4332272d2d623d3c1a52632a28387f0921130b3d7d2b52632b2f785c"
"3b3b43242f38297c31480d3b72284b3a0c60267b1a5471372165422 37d38"

Can you recover the key?

Hint: [check this if you do not have any idea]

Exercise MTV Cribs solution

Check solution by alex-bellon

Exercise 1.5 Understanding Cryptography

Compute the result without a calculator:

1. $15 \cdot 29 \bmod 13$
2. $2 \cdot 29 \bmod 13$
3. $2 \cdot 3 \bmod 13$
4. $-11 \cdot 3 \bmod 13$

Exercise 1.5 Understanding Cryptography: solution

Compute the result without a calculator:

1. $15 \cdot 29 \bmod 13 = 2 \cdot 3 \bmod 13 = 6 \bmod 13$
2. $2 \cdot 29 \bmod 13 = 2 \cdot 3 \bmod 13 = 6 \bmod 13$
3. $2 \cdot 3 \bmod 13 = 2 \cdot 3 \bmod 13 = 6 \bmod 13$
4. $-11 \cdot 3 \bmod 13 = 2 \cdot 3 \bmod 13 = 6 \bmod 13$

Exercise 1.8 Understanding Cryptography

What is the multiplicative inverse of 5 in:

1. $\mathbb{Z}_{11}$
2. $\mathbb{Z}_{12}$
3. $\mathbb{Z}_{13}$

Exercise 1.8 Understanding Cryptography: solution

What is the multiplicative inverse of 5 in:

1. $\mathbb{Z}_{11}$: 9
2. $\mathbb{Z}_{12}$: 5
3. $\mathbb{Z}_{13}$: 8

Exercise 1.9 Understanding Cryptography

Find x without a calculator:

1. $x = 3^2 \bmod 13$
2. $x = 7^2 \bmod 13$
3. $x = 3^{10} \bmod 13$
4. $x = 7^{100} \bmod 13$
5. $7^x = 11 \bmod 13$

Exercise 1.9 Understanding Cryptography: solution

Find x without a calculator:

1. $x = 3^2 \bmod 13 \equiv 9 \bmod 13$
2. $x = 7^2 \bmod 13 \equiv 49 \equiv 10 \bmod 13$
3. $x = 3^{10} \bmod 13 \equiv 9^5 \equiv 81^2 \cdot 9 \equiv 81 \equiv 3 \bmod 13$
4. $x = 7^{100} \bmod 13 \equiv 49^{50} \equiv 10^{50} \equiv (-3)^{10 \cdot 5} \equiv (-3^{10})^5$
   $\equiv (3^{10})^5 \equiv 3^5 \equiv 3^3 \cdot 3^2 \equiv 9 \bmod 13$
5. In general solving this problem is very hard (when the modulo is large).
   In our example, by trial:

$$7^5 = 11 \bmod 13$$

Exercise 6.5 Understanding Cryptography

Using the basic form of Euclid's algorithm, compute the greatest common divisor of:

1. 7469 and 2464
2. 2689 and 4001

For this problem use only a pocket calculator. Show every iteration step of Euclid's algorithm, i.e., don't write just the answer, which is only a number.

Exercise 6.5 Understanding Cryptography: solution

Solution (without step by step):

1. gcd(7469, 2464) = 77
2. gcd(4001, 2689) = 1

Exercise 6.6 Understanding Cryptography

Using the extended Euclidean algorithm, compute the greatest common divisor and the parameters s, t of:

1. 198 and 243
2. 1819 and 3587

For every problem check if $s*r_0 + t*r_1 = \gcd(r_0 , r_1)$ is actually fulfilled.

Exercise 6.6 Understanding Cryptography: solution

Solution (without step by step):

1. gcd(198, 243) = 9, s = 9, t = -11
2. gcd(3587, 1819) = 17, s = -36, t = 71

Exercise on EEA

Find the multiplicative inverse of 19 modulo 160 using EEA

Exercise on EEA: solution

[Solution] by Giuliano Abruzzo

Exercise 5.2 Understanding Cryptography

We consider known-plaintext attacks on block ciphers by means of an exhaustive key search where the key is k bits long. The block length counts n bits with n > k.

1. How many plaintexts and ciphertexts are needed to successfully break a block cipher running in ECB mode? How many steps are done in the worst case?
2. Assume that the initialization vector IV for running the considered block cipher in CBC mode is known. How many plaintexts and ciphertexts are now needed to break the cipher by performing an exhaustive key search? How many steps need now maximally be done? Briefly describe the attack.
3. How many plaintexts and ciphertexts are necessary, if you do not know the IV?
4. Is breaking a block cipher in CBC mode by means of an exhaustive key search considerably more difficult than breaking an ECB mode block cipher?

Exercise 5.2 Understanding Cryptography: solution

1. 1 pair, 2^k attempts

2. To perform an attack we need:
   a. a plaintext/ciphertext block pair (x_i, y_i)
   b. previous ciphertext block y_(i-1) or IV if i=0

   2^k attempts

3. Knowing the IV does reduce the number of attacks
4. Same asymptotic complexity wrt attack attempts. CBC is better to prevent other attacks (e.g., substitutions)

Exercise 5.3 Understanding Cryptography

In a company, all files which are sent on the network are automatically encrypted by using AES-128 in CBC mode. A fixed key is used, and the IV is changed once per day. The network encryption is file-based, so that the IV is used at the beginning of every file.

You managed to spy out the fixed AES-128 key, but do not know the recent IV.

Today, you were able to eavesdrop two different files, one with unidentified content and one which is known to be an automatically generated temporary file and only contains the value 0xFF. Briefly describe how it is possible to obtain the unknown initialization vector and how you are able to determine the content of the unknown file.

Exercise 5.3 Understanding Cryptography: solution

Since you know the key K and the pair (x0, y0) (from the first file), the unknown IV can easily be obtained by converting the equation:

$IV = d_K(y0) \oplus x0$

After that, the second (unidentified) file can easily be decrypted by using the decryption equation $x_i = dK\ (y_i) \oplus y_{i-1}$ (with $y_{-1} = IV$).

Exercise 5.5 Understanding Cryptography

Describe how the OFB mode can be attacked if the IV is not different for each execution of the encryption operation.

Exercise 5.5 Understanding Cryptography: solution

If the same IV is used for the OFB encryption, the confidentiality may be compromised. If a plaintext block x_j of such a message m is known, the output can be computed easily from the ciphertext block y_j of the message m. This information then allows the computation of the plaintext block x'_j of any other message m′ that is encrypted using the same IV.

Exercise 5.10 Understanding Cryptography

Sometimes error propagation is an issue when choosing a mode of operation in practice. In order to analyze the propagation of errors, let us assume a bit error (i.e., a substitution of a “0” bit by a “1” bit or vice versa) in a ciphertext block y_i.

1. Assume an error occurs during the transmission in one block of ciphertext, let’s say y_i. Which cleartext blocks are affected on Bob’s side when using the ECB mode?

2. Again, assume block y_i contains an error introduced during transmission. Which cleartext blocks are affected on Bob’s side when using the CBC mode?

3. Suppose there is an error in the cleartext x_i on Alice’s side. Which cleartext blocks are affected on Bob’s side when using the CBC mode?

4. Prepare an overview of the effect of bit errors in a ciphertext block for the modes ECB, CBC, CFB, OFB and CTR. Differentiate between random bit errors and specific bit errors when decrypting y_i.

Exercise 5.10 Understanding Cryptography: solution

1. only block x_i is affected
2. block x_i is corrupted entirely, block x_{i+1} is corrupted in a specific pattern
3. only block x_i is affected
4.

Mode	Effect of error in y_i (RBE: random bit errors, SBE: specific bit errors)
ECB	RBE in decryption of y_i
CBC	RBE in decryption of y_i , SBE in decryption of y_{i+1}
CFB	SBE in decryption of y_i , RBE in decryption of y_{i+1}
OFB	SBE in decryption of y_i
CTR	SBE in decryption of y_i

Exercise AES-ABC

Credits: PicoCTF 2019

Description: AES-ECB is bad, so I rolled my own cipher block chaining mechanism - Addition Block Chaining! You can find the source here: aes-abc.py.

Goal: explain how AES-ABC is not that different from AES-CBC

Exercise AES-ABC: solution

Idea: given a ciphertext you can always revert it to a ciphertext generated by AES-ECB

Solution by Dvd848

Exercise We three keys

Credits: SwapCTF

Description: a server is performing encryption using this code. Is it safe?

Exercise We three keys: solution

The server is doing two “wrong” things:

1. The key and the IV are the same
2. The IV is kept constant for different messages

Solution:

- write-up by Mathis Hammel
- write-up by xrmon

Exercise Reverse Search Algorithm

Credits: CTF HSCTF 6

Input:

n = 561985565696052620466091856149686893774419565625295691069663316673425409620917583731032457879432617979438142137

e = 65537

c = 32805527921212861689820380998303970878749038465072589074857692720888305538143000075662436963682090370477583577 7

Goal: decrypt the ciphertext c

Exercise Reverse Search Algorithm: hint #1

Idea: try to factor n with small prime numbers

Exercise Reverse Search Algorithm: solution

The number n has a small prime factor equal to 29. Assuming the cipher is RSA, we can compute the decryption key d and then decrypt the ciphertext.

Script: solve.py

Credits: kurawa2

Exercise Massive RSA

Credits: CTF HSCTF 6

Input: massive.txt

Goal: decrypt the ciphertext c

Exercise Massive RSA: hint

Idea: what happens if n is a prime?

Exercise Massive RSA: solution

The number n is a prime. Assuming the cipher is RSA, we can compute the decryption key d very easily and then decrypt the ciphertext.

Script: solve.py

Credits: kurawa2

Exercise Really Secure Algorithm

Credits: CTF HSCTF 6

Input: secure.txt

Goal: decrypt the ciphertext c

Exercise Really Secure Algorithm: hint

Idea: what happens if n is the square of a prime?

Exercise Really Secure Algorithm: solution

The number n is the square of prime number p:

16225510719965861964299051658340559066224635411075742500953901749924501886090804067406052688894869028683583501052917637552385089084807531319036985272636554557876754514524927502408114799014949174520357440885167280739363628642463479075654764698947461583766215118582826142179234382923872619079721726020446020581078274482268162477580369246821166693123724514271177264591824616458410293414647

Assuming the cipher is RSA, we can compute the decryption key d very easily and then decrypt the ciphertext.

Script: solve.py

Credits: kurawa2

Exercise DHKE with three parties

Goal: Discuss how DHKE can be extended to work with three users (Alice, Bob, and Sempronio) instead of just two (Alice and Bob). Discuss possible attacks against your proposal (passive vs active attacker).

Exercise DHKE with three parties: solution

See one solution at this wikipedia page.