# 🔐 Tutorial 4: Symmetric Encryption

In this tutorial, we will explore some foundational concepts in **historical cryptographic schemes** and **symmetric cryptography**.
Recall that in **symmetric key encryption**, the same secret key is used for both the encryption and decryption processes.

## 🧩 1. Revisiting Simple Encryption

You are given a text file named **enc.txt** by a friend. They say they have no idea what it means—after all, they've never taken a class in cryptography (and they didn't enjoy history much in school either).

They watch you in wonder and awe as you begin decrypting it using Python, without relying on any cryptography libraries.

Your task is to **decrypt the given file `enc.txt`**.

> **Recall:** This is a *ciphertext-only attack* scenario, where you are given only the encrypted message (ciphertext) and no information about the plaintext or the key. Your job is to analyze the ciphertext and deduce the plaintext through cryptanalysis techniques.

### a) Observe the contents of the given file

Write code that prints the contents of the file.

In [1]:
```
file = open("enc.txt")     # TODO: Open the enc.txt file
content = file.read()      # TODO: Read the contents
print(content)             # TODO: Print the contents
file.close()               # Close the file
```

```
13011811512811513711912913711112413711112813013711913013712911113512913712
```

```
119124137135125131128137111291301251241191291181231151241301371351251311
713311912212213712311912912913713011811513711712812513112411413711312512
261221151301151221351371111241141371281151231111191241371121251121121191
117137120131129130137111137116115133137119124113118115129137111121251321
513711913013711912413713311811113013712311911711813013712911511512313713
251371121151371111371291221191171181301221351371161251251221191291181371
111124124115128137130118119129137119129137111137123125123115124130137116
512813712913112611512811213711112411413711411512211911311113011513711312
241131151241301281111301191251241371121251121371111241141371161221251111
137116122125111130137111124114137112125112137119117124125128115137111122
213711312512412911911411512811113011912512413712511613713512513112813712
331241371331151191171181301371291191231261221351371221151301371351251311
129115122116137133111116130137118119117118115128137114125137124125130137
211912913011512413713012513713311811113013711112413511212511413513712912
351291371301251371351251311371111301371301181191291371261251191241301371
115113111131129115137130118115135137111128115137131124122119121115122135
713012513712911113513711112413513011811912411713711811512212611613112213
301181151351371111281151371231251291301371221191211151221351371301251371
111135137129125123115130118119124117137111122125124117137130118115137122
912411512913712511613711712512511413711712511413713512513113711311112413
371261251291291191121221351371121151371161221351191241171371191301371191
137132119130111122122135137119123126125128130111124130137124125130137130
513711211512211911513211513713011811512313712512813713011811513513713311
221221371291311141141151241222135137112115137128119117118130137133111116
137118119117118115128137111124114137118119117118115128137130128135137111
711611513313712913312512512612913711711512413012211513712512411512913711
301371161191281291301371301181151241371141281191161301371111121251321151
130118115137130128115115130125126129137112128115111130118119124117137128
511713112211112812213513711412513712412513013713311113211513711113013711
241351121251141351371371331181151241371351251311371181111321151371141251
115137130118119129137111137116115133137130119123115129137135125131137133
912212213711611912411413713011811513712312512311512413013712511613711411
291301281111131301191251241371281111261191141221351371151111291191151281
111124114137115111291191151281371301251371111131181191151321151371371351
513113713311912212213713011811512413712211511112812413711112212213712912
513113713311912212213713011811512413712211511112812413711112212213712912
281301291371251161371301181191241171291371111121251311301371181251331371
125137113125124130128125122137135125131128137116122119117118130137135125
112813712912611511511413713512513112813712311112411513113211512811111211
221191301351371111241141371301181151371301281191131211371311291311111221
135137122119115129137119124137124125130137130118119124121119124117137130
512513711811112811413711111212513113013713311811113011513211512813713512
311371331111241301371301251371141251371121311301371201311291301371111221
125133119124117137119130137130125137118111126126115124137111129137119116
711913013713311512811513711712511912411713713012513711112413513311113513
371351251311371331191221221371111221291251371221151111281241371111121251
130137118125133137130125137122111124114137126128125126115128122135137133
811911311813711912913712912512311513011811912411713713512513113713313312
221371111221231251291301371131151281301111191241221351371291131281151331
131126137111124114137129113128115133137131126137112111114122135137125124
713512513112813711611912812913013711113013011512312613013713713011811512
151371111281151371261281191321111301151371131221311121291371351251311371
111124137120125119124137133118119113118137118115122126137135125131137113
311811911513211513713011811513711112221221912312612512813011112413013712
312311512413013712511613711411912913012811111313011912512413713011811513
137118119128115137126115125126122115137133119130118137129131128126128119
911912411713711212511411911512913712512813712512611912411912512412913713
251371221151111261371251311301371161281251231371121151181191241141371121
129118115129137111124114137115134118119112119130137111124114137125128137
513412612211111912413713011811512313711113013713011811513711312811913011
131111221371231251231151241301291371161151331371171151241311191241151371
```

```
119130113118118119121115128129137133119122122137112115137111112122115137130125137111116116125128114137130125137120125119124137130118115129115137113122113112912937112131130137129125123115137123111135137112115137111112122115137130125137117115130137130115123126125128111128135137115123126122125135123115124130137111130137130118115123
```

## b) Check the properties of the ciphertext

That doesn't reveal much, does it?

Let's examine some properties of the ciphertext:

- Check the length of the ciphertext.
- Find the factors of its length (use the **factorint** function in **sympy**) – Reference
- Which numbers divide the length?

In [3]:
```
from sympy.ntheory import factorint

print(len(content)) ## TODO print the total length of the content
factorint(len(content)) ## TODO use the factorint function to find the pr
```

```
9210
```

Out[3]:
```
{2: 1, 3: 1, 5: 1, 307: 1}
```

Still not much insight. Let's perform some alphabet/symbol analysis:

- Generate frequency plots assuming symbol lengths from 1 to 5.
- Write the code in a generic way:
  - Allow input of the symbol length to consider.
  - The code should progressively identify symbols, store them in a dictionary, and track each symbol's frequency.
  - Once the dictionary is fully built, plot the frequency histogram of the symbols.
  - Based on the plots, which symbol length do you think is the correct one?

In [ ]:
```
## TODO: Write a function to build the symbol frequency dictionary
## TODO: It will take the filename and the symbol length as inputs

def build_symbol_dictionary(filename, symbol_length):
    file = open(filename, 'r')  ## TODO: Open the given file
    count = {}                  ## TODO: Build an empty dictionary
    symbol = ""                 ## Placeholder for the symbols - no need

    while True:
        # TODO: Read the file by symbol_length
        symbol = file.read(symbol_length)

        if not symbol:          ## TODO: If there is no symbol, it means
            file.close()
            return count
        else:
            if symbol in count: ## TODO: If the current symbol is already
                count[symbol] += 1
            else:               ## TODO: Else, add the symbol to the dicti
                count[symbol] = 1
```

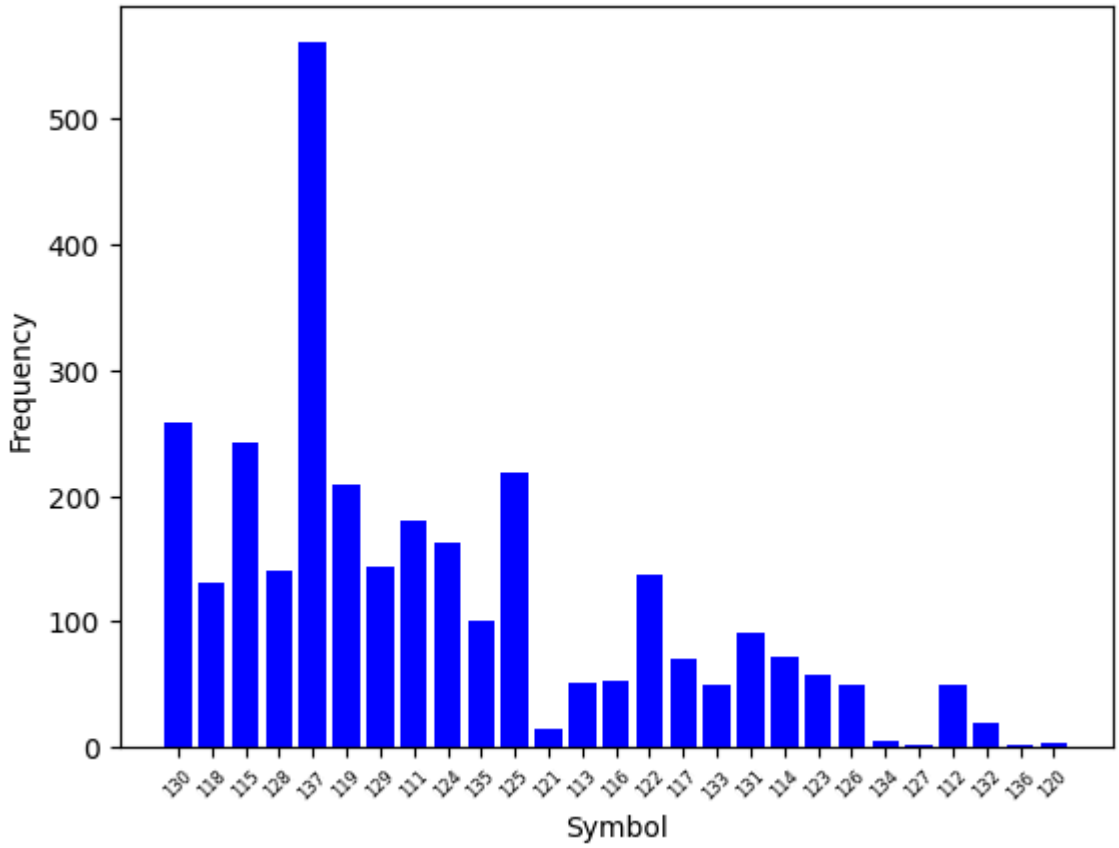
In [12]:
```
symbol_frequency = build_symbol_dictionary("enc.txt", 3)

print(symbol_frequency.keys())
print(sum(symbol_frequency.values()))
```

```
dict_keys(['130', '118', '115', '128', '137', '119', '129', '111', '124',
'135', '125', '121', '113', '116', '122', '117', '133', '131', '114', '12
3', '126', '134', '127', '112', '132', '136', '120'])
3070
```

In [13]:
```
import matplotlib.pyplot as plt

plt.bar(list(symbol_frequency.keys()), symbol_frequency.values(), color='
plt.xticks(rotation = 45, size =6)
plt.xlabel("Symbol")
plt.ylabel("Frequency")
plt.show()
```




## c) Decide on the symbol length

From the above analysis decide on the symbol length. Now we need to try some substitutions

- What symbol can be the whitespace?
- What symbol can be the letter 'e'?
- Write a code to try out a few substitutions and see what makes sense. Try several runs with different guesses.

In [14]:
```
file = open("enc.txt", 'r')

content=file.read()
```

```
sub_content=""

for i in range(0,len(content),3):
    symbol = content[i:i+3]
    if symbol=="137": ##TODO Experiment with guessing some symbols. You c
        symbol=" "
    if symbol=="115": ##TODO Experiment with guessing some symbols. You c
        symbol="e"
    sub_content = sub_content+symbol
print(sub_content)

file.close()
```

130118e128e 119129 111124 111128130 119130 129111135129 125128 12811113011
8e128 111 121124111113121 130125 116122135119124117 130118e 12112411111312
1 122119e129 119124 122e111128124119124117 118125133 130125 13011812812513
3 135125131128129e122116 111130 130118e 117128125131124114 111124114 12311
9129129 126119113121 111 124119113e 114111135 130118e 11811913011311811811
9121e128129 117131119114e 130125 130118e 117111122111134135 129131117117e1
29130129 111124114 130128135 119130 130118e 116119128129130 126111128130 1
19129 e111129135 111122122 119130 128e127131119128e129 119129 129119123126
122135 130118e 111112119122119130135 130125 130118128125133 13512513112812
9e122116 116125128133111128114 133119130118 111122122 135125131128 133e119
117118130 111124114 130118e 133119122122119124117124e129129 124125130 1301
25 123119124114 130118111130 119130129 117125119124117 130125 118131128130
130118111130 119129 119130129 117125119124117 130125 118131128130 119116 1
35125131 116111119122 130125 123119129129 130118e 117128125131124114 12312
5129130 126e125126122e 116111119122 130125 123119129129 130118e 1171281251
31124114 111124114 119116 130118e135 111128e 128e111122122135 130128135119
124117 126128125126e128122135 130118e 122119121e122119118125125114 119129
130118111130 130118e135 133119122122 116111119122 130125 123119129129 1191
30 116111119128122135 118111128114 113122e111128122135 119130 119129 13011
8e 129e113125124114 126111128130 130118e 123119129129119124117 13311811911
3118 126128e129e124130129 130118e 114119116116119113131122130119e129 12512
4e 126128125112122e123 119129 130118111130 135125131 118111132e 130125 123
119129129 130118e 117128125131124114 111113113119114e124130111122122135 11
9130129 124125 117125125114 114e122119112e128111130e122135 119124130e12411
4119124117 130125 123119129129 130118e 117128125131124114 112e113111131129
e 135125131 133125124130 135125131 118111132e 130125 118111132e 1351251311
28 111130130e124130119125124 129131114114e124122135 11411912913012811111131
30e114 112135 129125123e130118119124117 e122129e 133118e124 135125131128e
118111122116133111135 130118e128e 129125 130118111130 135125131 111128e 12
4125 122125124117e128 130118119124121119124117 111112125131130 11611112212
2119124117 125128 111112125131130 130118e 117128125131124114 125128 111112
125131130 118125133 123131113118 119130129 117125119124117 130125 11813112
8130 119116 135125131 116111119122 130125 123119129129 119130 119130 11912
9 124125130125128119125131129122135 114119116116119113131122130 130125 126
128119136e 135125131128 111130130e124130119125124 111133111135 11612812512
3 130118e129e 130118128ee 130118119124117129 114131128119124117 130118e 12
9126122119130 129e113125124114 135125131 118111132e 111130 135125131128 11
4119129126125129111122 118e124113e 123125129130 126e125126122e129 11611111
9122131128e 111124114 130118e119128 e132e124130131111122 11411912911912212
2131129119125124123e124130 133119130118 130118119129 e13411811912211112811
1130119124117 111124114 129126e113130111113131122111128 129126125128130 11
9116 118125133e132e128 135125131 111128e 122131113121135 e124125131117118
130125 118111132e 135125131128 111130130e124130119125124 123125123e1241301
11128119122135 114119129130128111113130e114 111130 130118e 113128131113119
111122 123125123e124130 112135 129111135 111 117125128117e125131129 126111
119128 125116 122e117129 125128 111 112125123112 117125119124117 125116116
119124 135125131128 132119113119124130135 125128 112135 129131114114e12412
2135 129126125130130119124117 111124 e134130128e123e122135 128111128e 1291
26e113119e129 125116 112ee130122e 113128111133122119124117 111122125124117
111 124e111128112135 130133119117 130118e124 119124 135125131128 111129130
125124119129118123e124130 135125131 133119122122 123119129129 130118e 1171
28125131124114 113125123126122e130e122135 111124114 128e123111119124 11212
5112112119124117 120131129130 111 116e133 119124113118e129 111112125132e 1
19130 119124 133118111130 123119117118130 129ee123 130125 112e 111 1291221
19117118130122135 116125125122119129118 123111124124e128 130118119129 1191
29 111 123125123e124130 116125128 129131126e128112 111124114 114e122119113
111130e 113125124113e124130128111130119125124 112125112 111124114 11612212
5111130 116122125111130 111124114 112125112 119117124125128e 111122122 113
125124129119114e128111130119125124 125116 135125131128 125133124 133e11911
7118130 129119123126122135 122e130 135125131128129e122116 133111116130 118

```
119117118e128 114125 124125130 122119129130e124 130125 133118111130 111124
135112125114135 129111135129 130125 135125131 111130 130118119129 12612511
9124130 112e113111131129e 130118e135 111128e 131124122119121e122135 130125
129111135 111124135130118119124117 118e122126116131122 130118e135 111128e
123125129130 122119121e122135 130125 129111135 129125123e130118119124117 1
11122125124117 130118e 122119124e129 125116 117125125114 117125114 1351251
31 113111124130 126125129129119112122135 112e 116122135119124117 119130 11
9129 132119130111122122135 119123126125128130111124130 124125130 130125 11
2e122119e132e 130118e123 125128 130118e135 133119122122 129131114114e12412
2135 112e 128119117118130 133111116130 118119117118e128 111124114 11811911
7118e128 130128135 111 116e133 129133125125126129 117e124130122e 125124e12
9 111130 116119128129130 130118e124 114128119116130 111112125132e 130118e
130128ee130125126129 112128e111130118119124117 128e1171311221111128122135 1
14125 124125130 133111132e 111130 111124135112125114135  133118e124 135125
131 118111132e 114125124e 130118119129 111 116e133 130119123e129 135125131
133119122122 116119124114 130118e 123125123e124130 125116 1141191291301281
11113130119125124 128111126119114122135 e111129119e128 111124114 e11112911
9e128 130125 111113118119e132e  135125131 133119122122 130118e124 122e1111
28124 111122122 129125128130129 125116 130118119124117129 111112125131130
118125133 130125 113125124130128125122 135125131128 116122119117118130 135
125131128 129126ee114 135125131128 123111124e131132e128111112119122119130 1
35 111124114 130118e 130128119113121 131129131111122122135 122119e129 1191
24 124125130 130118119124121119124117 130125125 118111128114 1111121251311
30 133118111130e132e128 135125131 133111124130 130125 114125 112131130 120
131129130 111122122125133119124117 119130 130125 118111126126e124 111129 1
19116 119130 133e128e 117125119124117 130125 111124135133111135  135125131
133119122122 111122129125 122e111128124 111112125131130 118125133 130125 1
22111124114 126128125126e128122135 133118119113118 119129 129125123e130118
119124117 135125131 133119122122 111122123125129130 113e128130111119124122
135 129113128e133 131126 111124114 129113128e133 131126 112111114122135 12
5124 135125131128 116119128129130 111130130e123126130  130118e128e 111128e
126128119132111130e 113122131112129 135125131 113111124 120125119124 13311
8119113118 118e122126 135125131 111113118119e132e 130118e 1111221221191231
26125128130111124130 123125123e124130 125116 1141191291301281111113130119125
5124 130118e135 118119128e 126e125126122e 133119130118 1291311281261281191
29119124117 112125114119e129 125128 125126119124119125124129 130125 122e11
1126 125131130 116128125123 112e118119124114 112131129118e129 111124114 e1
34118119112119130 111124114 125128 e134126122111119124 130118e123 111130 1
30118e 113128119130119113111122 123125123e124130129 116e133 117e1241311191
24e 118119130113118118119121e128129 133119122122 112e 111112122e 130125 11
1116116125128114 130125 120125119124 130118e129e 113122131112129 112131130
129125123e 123111135 112e 111112122e 130125 117e130 130e123126125128111128
135 e1231261222125135123e124130 111130 130118e123
```

Hopefully by now you have identified some correct mappings. You can continue this process to find all the mappings, but it can be a bit tedious. Let's see if we can identify a pattern:

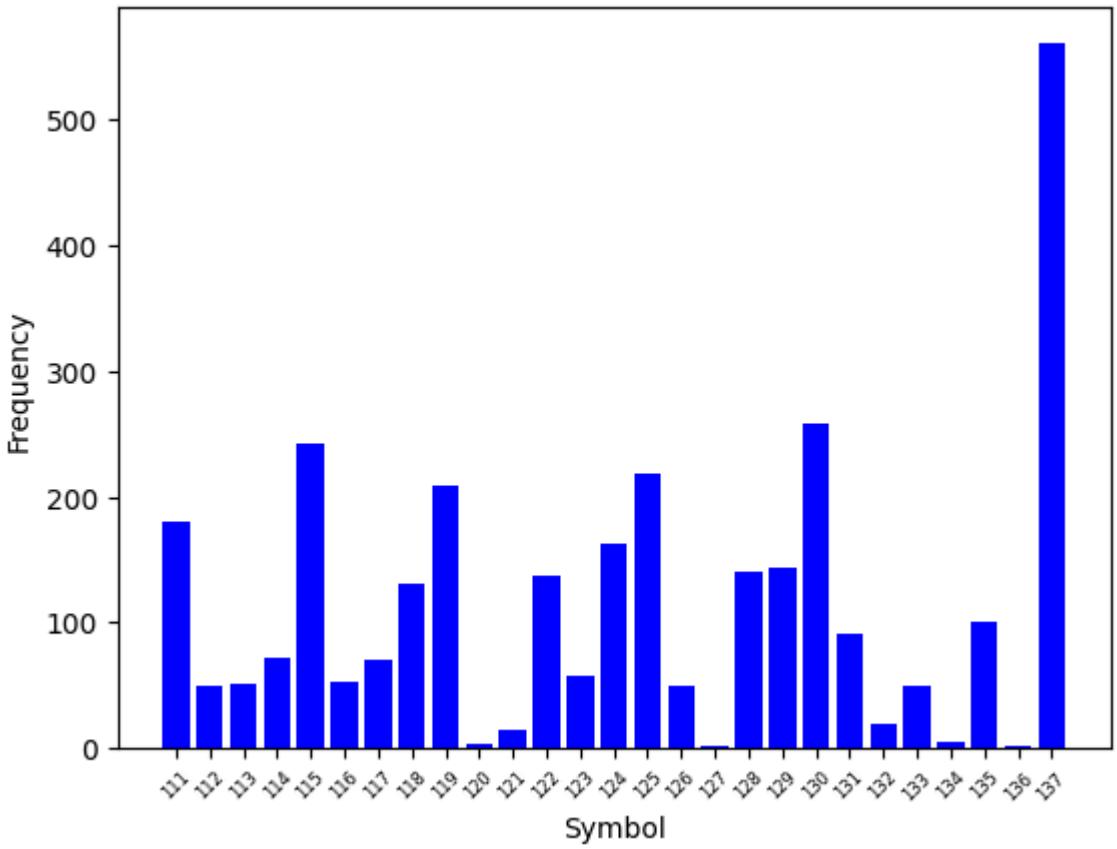
- Build a new sorted dictionary.
- Replot the symbol histogram (this time sorting the keys).

In [15]:
```
keys = list(symbol_frequency.keys())
keys.sort()
sorted_dict = {i: symbol_frequency[i] for i in keys}

import matplotlib.pyplot as plt

plt.bar(list(sorted_dict.keys()), sorted_dict.values(), color='b')
plt.xticks(rotation = 45, size =6)
```

```
plt.xlabel("Symbol")
plt.ylabel("Frequency")
plt.show()
```




Based on this graph and the substitutions you found earlier, you can now express the mapping as a formula.

- What is the equation for the mapping?

> **Solution:**
>
> The mapping can be expressed as:
> [ c = p + 111 ] where (p=0) corresponds to 'a', (p=1) to 'b', and so on. The value 137 represents the whitespace character.

### d) Write a code to decipher the full text.

In [16]:
```
import string

chars = list(string.ascii_lowercase)

file = open("enc.txt", 'r')

content = file.read()
file.close()

plain_content = ""

for i in range(0, len(content), 3):
```

```
    symbol = content[i:i+3]
    if symbol != '137':  # TODO: Write the program logic for non-whitespa
        new_symbol = int(symbol) - 111
        new_symbol = chars[new_symbol]
        plain_content += new_symbol
    else:
        plain_content += " "

print(plain_content)

with open("plain.txt", "w") as text_file:
    n = text_file.write(plain_content)
```

```
there is an art it says or rather a knack to flying the knack lies in lear
ning how to throw yourself at the ground and miss pick a nice day the hitc
hhikers guide to the galaxy suggests and try it the first part is easy all
it requires is simply the ability to throw yourself forward with all your
weight and the willingness not to mind that its going to hurt that is its
going to hurt if you fail to miss the ground most people fail to miss the
ground and if they are really trying properly the likelihood is that they
will fail to miss it fairly hard clearly it is the second part the missing
which presents the difficulties one problem is that you have to miss the g
round accidentally its no good deliberately intending to miss the ground b
ecause you wont you have to have your attention suddenly distracted by som
ething else when youre halfway there so that you are no longer thinking ab
out falling or about the ground or about how much its going to hurt if you
fail to miss it it is notoriously difficult to prize your attention away f
rom these three things during the split second you have at your disposal h
ence most peoples failure and their eventual disillusionment with this exh
ilarating and spectacular sport if however you are lucky enough to have yo
ur attention momentarily distracted at the crucial moment by say a gorgeou
s pair of legs or a bomb going off in your vicinty or by suddenly spotting
an extremely rare species of beetle crawling along a nearby twig then in y
our astonishment you will miss the ground completely and remain bobbing ju
st a few inches above it in what might seem to be a slightly foolish manne
r this is a moment for superb and delicate concentration bob and float flo
at and bob ignore all consideration of your own weight simply let yourself
waft higher do not listen to what anybody says to you at this point becaus
e they are unlikely to say anything helpful they are most likely to say so
mething along the lines of good god you cant possibly be flying it is vita
lly important not to believe them or they will suddenly be right waft high
er and higher try a few swoops gentle ones at first then drift above the t
reetops breathing regularly do not wave at anybody  when you have done thi
s a few times you will find the moment of distraction rapidly easier and e
asier to achieve  you will then learn all sorts of things about how to con
trol your flight your speed your maneuverability and the trick usually lie
s in not thinking too hard about whatever you want to do but just allowing
it to happen as if it were going to anyway  you will also learn about how
to land properly which is something you will almost certainly screw up and
screw up badly on your first attempt  there are private clubs you can join
which help you achieve the allimportant moment of distraction they hire pe
ople with surprising bodies or opinions to leap out from behind bushes and
exhibit and or explain them at the critical moments few genuine hitchhiker
s will be able to afford to join these clubs but some may be able to get t
emporary employment at them
```

## 🔑 2. Doing it Right — Encryption

Once you have decrypted `enc.txt`, you decide to do it properly this time.

- Write code to read the decoded text file and encrypt it using AES-CBC.
- Use the `pycrypto` library: https://pycryptodome.readthedocs.io/en/latest.
- You might need to pad your data. Why?
- Choose a key length of 128 bits.
- Generate an IV using `Crypto.Random`.
- Print the output as a hex-encoded string.
- Write the output as binary to `text.aes`.

In [17]:
```
# from os import lseek  # (Unused import)
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad

def AES_Encrypt(inFileName, outFileName):
    # TODO: Generate random 16-byte key and IV
    key = get_random_bytes(16)
    iv = get_random_bytes(16)

    cipher = AES.new(key, AES.MODE_CBC, iv)

    # TODO: Load the input file
    with open(inFileName, 'rb') as text_file:
        plaintext = text_file.read()

    # TODO: Perform the encryption
    ciphertext = cipher.encrypt(pad(plaintext, 16))

    # TODO: Store the encrypted file
    with open(outFileName, "wb") as cipher_file:
        cipher_file.write(ciphertext)

    # TODO: Return the key and initialization vector (needed for decrypti
    return key, iv

key, iv = AES_Encrypt("plain.txt", "testing.enc")
```

## 🔑 3. Doing it Right — Decryption

Write a code snippet to check whether you can decrypt the file you just encrypted.

In [18]:
```
from Crypto.Util.Padding import unpad

def AES_Decrypt(key, iv, inFileName):
    cipher = AES.new(key, AES.MODE_CBC, iv)

    # TODO: Load the encrypted file
    with open(inFileName, 'rb') as text_file:
        ciphertext = text_file.read()

    # TODO: Perform the decryption
    plaintext = unpad(cipher.decrypt(ciphertext), 16)

    print(plaintext)

AES_Decrypt(key, iv, "testing.enc")
```

```
b'there is an art it says or rather a knack to flying the knack lies in le
arning how to throw yourself at the ground and miss pick a nice day the hi
tchhikers guide to the galaxy suggests and try it the first part is easy a
ll it requires is simply the ability to throw yourself forward with all yo
ur weight and the willingness not to mind that its going to hurt that is i
ts going to hurt if you fail to miss the ground most people fail to miss t
he ground and if they are really trying properly the likelihood is that th
ey will fail to miss it fairly hard clearly it is the second part the miss
ing which presents the difficulties one problem is that you have to miss t
he ground accidentally its no good deliberately intending to miss the grou
nd because you wont you have to have your attention suddenly distracted by
something else when youre halfway there so that you are no longer thinking
about falling or about the ground or about how much its going to hurt if y
ou fail to miss it it is notoriously difficult to prize your attention awa
y from these three things during the split second you have at your disposa
l hence most peoples failure and their eventual disillusionment with this
exhilarating and spectacular sport if however you are lucky enough to have
your attention momentarily distracted at the crucial moment by say a gorge
ous pair of legs or a bomb going off in your vicinty or by suddenly spotti
ng an extremely rare species of beetle crawling along a nearby twig then i
n your astonishment you will miss the ground completely and remain bobbing
just a few inches above it in what might seem to be a slightly foolish man
ner this is a moment for superb and delicate concentration bob and float f
loat and bob ignore all consideration of your own weight simply let yourse
lf waft higher do not listen to what anybody says to you at this point bec
ause they are unlikely to say anything helpful they are most likely to say
something along the lines of good god you cant possibly be flying it is vi
tally important not to believe them or they will suddenly be right waft hi
gher and higher try a few swoops gentle ones at first then drift above the
treetops breathing regularly do not wave at anybody  when you have done th
is a few times you will find the moment of distraction rapidly easier and
easier to achieve  you will then learn all sorts of things about how to co
ntrol your flight your speed your maneuverability and the trick usually li
es in not thinking too hard about whatever you want to do but just allowin
g it to happen as if it were going to anyway  you will also learn about ho
w to land properly which is something you will almost certainly screw up a
nd screw up badly on your first attempt  there are private clubs you can j
oin which help you achieve the allimportant moment of distraction they hir
e people with surprising bodies or opinions to leap out from behind bushes
and exhibit and or explain them at the critical moments few genuine hitchh
ikers will be able to afford to join these clubs but some may be able to g
et temporary employment at them'
```

## 🕵️ 4. Ciphertext Tampering in CBC Mode

AES in CBC (Cipher Block Chaining) mode links each ciphertext block to the previous one. See the given figure below.

 CBC

Now consider the following:

You have a ciphertext encrypted using AES-CBC.

One byte of a ciphertext block is corrupted (e.g., flipped or modified).

**Questions:**

1. What will happen to the decryption of the corrupted block?
2. What will happen to the next block during decryption?
3. Will the remaining blocks (after the next one) decrypt correctly or incorrectly? Why?

**Solution: Ciphertext Tampering in CBC Mode**

1. **Decryption of the corrupted block:**
   The corrupted ciphertext block will decrypt to garbage because the corrupted byte causes the entire decrypted block to become invalid and appear as random noise.
2. **Effect on the next block during decryption:**
   The next plaintext block will have exactly one byte corrupted at the same position as the corrupted ciphertext byte, due to the XOR operation with the corrupted ciphertext block. The rest of the bytes in the next plaintext block will decrypt correctly.
3. **Remaining blocks after the next one:**
   All subsequent plaintext blocks will decrypt correctly since they depend on uncorrupted ciphertext blocks beyond the corrupted one.