

Maria Valencia

Csc 154

Lab 2

2.1 Encoding and decoding

This encoding pattern is from base64.

The screenshot shows a web-based Base64 encoding tool. On the left, under the 'Recipe' tab, the 'From Base64' section is active. It features a dropdown menu for 'Alphabet' with the value 'A-Za-z0-9+/' selected. Below this, there is a checked checkbox for 'Remove non-alphabet chars' and an unchecked checkbox for 'Strict mode'. On the right, the 'Input' field contains the text 'WW91IGhhY2t1ciB5b3UhIQ=='. Below the input, a status bar shows 'REC 26' and a list icon. The 'Output' field displays the decoded text 'You hacker you!!'.

This encoding pattern is from Decimal.

The screenshot shows a web-based Decimal encoding tool. On the left, under the 'Recipe' tab, the 'From Decimal' section is active. It has a 'Delimiter' dropdown set to 'Space' and an unchecked checkbox for 'Support signed values'. On the right, the 'Input' field contains a multi-line decimal sequence: '69 110 99 111 100 105 110 103 32 105 115 32 110 111 116 32 101 110 99 114 121 112 116 105 111 110 32 58 41sdn'. A status bar below the input shows 'REC 112', a list icon, and 'Raw Bytes' with a 'CRLF (detected)' indicator. The 'Output' field shows the text 'Encoding is not encryptio :)'.

This encoding pattern is from Hex.

The screenshot shows a web-based Hex encoding tool. On the left, under the 'Recipe' tab, the 'From Hex' section is active. It has a 'Delimiter' dropdown set to 'Auto'. On the right, the 'Input' field contains the hex string '77 30 30 74 20 77 30 30 74'. A status bar below the input shows 'REC 26' and a list icon. The 'Output' field displays the decoded text 'woot woot'.

This encoding pattern is from Hex.

The screenshot shows the CyberChef interface with the 'From Hex' recipe selected. The input field contains a hex string: 48 65 78 20 69 73 20 63 6f 6d 6d 6f 6e 6c 79 20 75 73 65 64 20 77 69 74 68 20 61 73 73 65 6d 62 6c 7921. The output field displays the decoded text: Hex is commonly used with assembly!.

This encoding pattern is from Binary.

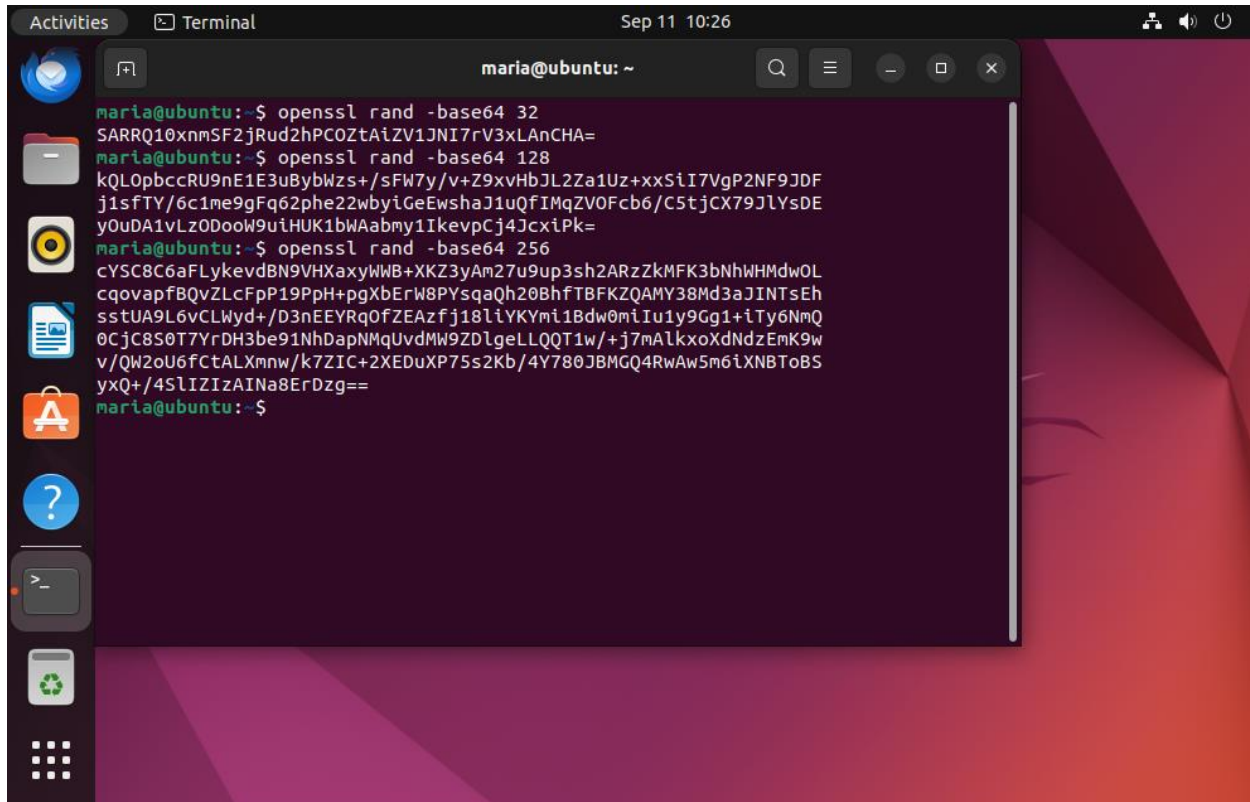
The screenshot shows the CyberChef interface with the 'From Binary' recipe selected. The input field contains a binary string: 01101111 01101110 01100101 00100111 01110011 00100000 01100001 01101110 01100100 00100000 01111010 01100101 01110010 01101111 00100111 01110011. The output field displays the decoded text: one's and zero's.

This encoding pattern is from base32 (step 3).

The screenshot shows the CyberChef interface with the 'To Base32' recipe selected. The input field contains the text: Cyber Chef is an awesome tool!. The output field displays the encoded text: IN4WEZLSEBBWQZLGEBUXGIDBNYQGC53FONXW2ZJAORXW63BBBUFA2CQ=.

2.2 Key Space

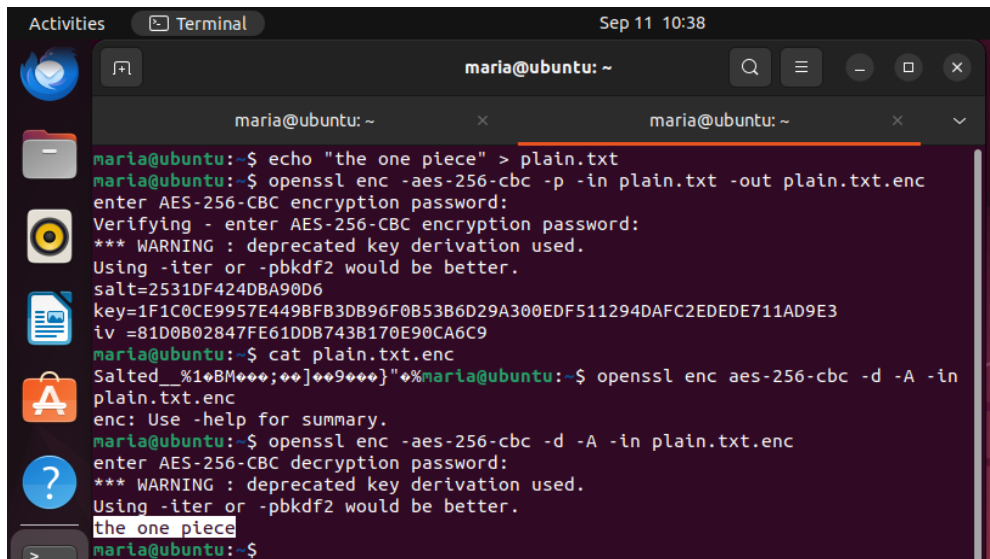
In this exercise, O opened OpenSSL in ubuntu and created a random 32, 128 and 256 key using the commands provided (openssl rand -base64 #).



```
Activities Terminal Sep 11 10:26
maria@ubuntu: ~
maria@ubuntu:~$ openssl rand -base64 32
SARRQ10xnmSF2jRud2hPC0ZtAiZV1JN17rV3xLANCHA=
maria@ubuntu:~$ openssl rand -base64 128
kQL0pbccRU9nE1E3uBybwzs+/sFW7y/v+Z9xvHbJL2Za1Uz+xxSiI7VgP2NF9JDF
j1sfTY/6c1me9gFq62phe22wbyiGeEwshaJ1uQfIMqZV0Fcb6/C5tjCX79JLYsDE
y0uDA1vLz0DooW9uiHUK1bWAabmy1IkevpCj4JcxiPk=
maria@ubuntu:~$ openssl rand -base64 256
cYSC8C6aFLykevdBN9VHXaxyWwB+XKZ3yAm27u9up3sh2ARzZkMFK3bNhwHMDwOL
cqovapfBQvZLcFpP19PpH+pgXbErW8PYsqaQh20BhfTBfKZQAMy38Md3aJINTsEh
sstUA9L6vCLWyd+/D3nEEYRqOfZEAzfj18liYKYmi1Bdw0mliu1y9Gg1+iTy6NmQ
0CjC8S0T7YrDH3be91NhDapNMqUvdMW9ZDlgeLLQQT1w/+j7mAlkxoXdNdzEmK9w
v/QW2oU6fCtALXmnw/k7ZIC+2XEDuXP75s2Kb/4Y780JBMGQ4RwAw5m6iXNBToBS
yxQ+/4SLIZIZAINa8ErDzg==
maria@ubuntu:~$
```

2.3 Symmetric Encryption

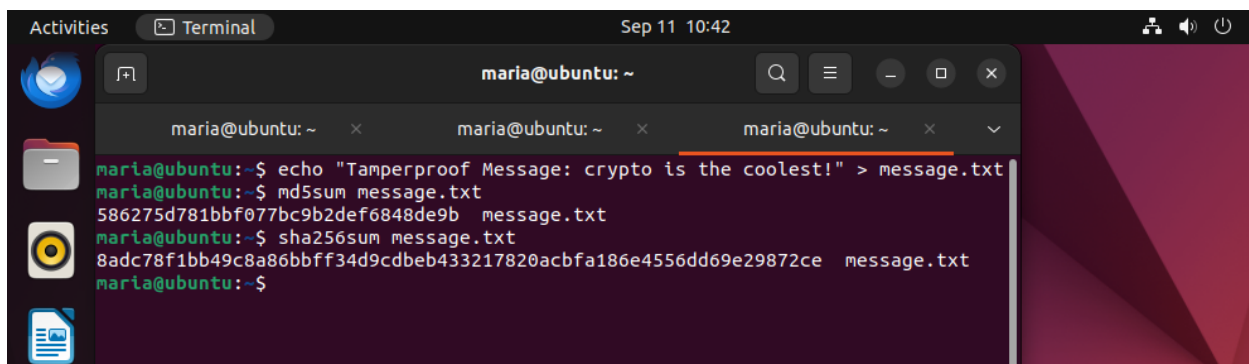
I used OpenSSL on Ubuntu to encrypt and decrypt using symmetric encryption. I did all three steps. Step 1 was to create a plain text file with a secret message. Step 2 was to encrypt that message using aes 256 encryption code block cipher mode. Step 3 was to decrypt the encrypted message using the key from step 1.

A terminal window on Ubuntu showing the process of encrypting and decrypting a file. The user first creates a file 'plain.txt' with the text 'the one piece'. Then, they use 'openssl enc -aes-256-cbc -p -in plain.txt -out plain.txt.enc' to encrypt it, providing a password. The terminal shows the encryption details, including a salt and key. Next, they use 'cat plain.txt.enc' to view the encrypted file. Finally, they use 'openssl enc -aes-256-cbc -d -A -in plain.txt.enc' to decrypt it, again providing the password, and the original text 'the one piece' is recovered.

```
maria@ubuntu:~$ echo "the one piece" > plain.txt
maria@ubuntu:~$ openssl enc -aes-256-cbc -p -in plain.txt -out plain.txt.enc
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=2531DF424DBA90D6
key=1F1C0CE9957E449BF83DB96F0B53B6D29A300EDF511294DAFC2EDEDE711AD9E3
iv =81D0B02847FE61DDB743B170E90CA6C9
maria@ubuntu:~$ cat plain.txt.enc
Salted__%1BM***;***]9***}"%maria@ubuntu:~$ openssl enc -aes-256-cbc -d -A -in
plain.txt.enc
enc: Use -help for summary.
maria@ubuntu:~$ openssl enc -aes-256-cbc -d -A -in plain.txt.enc
enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
the one piece
maria@ubuntu:~$
```

2.4 Hash Generation

I created a message in a new file to be used with hashing utilities on Ubuntu and then entered the commands given and noticed the difference of each output.

A terminal window on Ubuntu showing the generation of MD5 and SHA256 hashes for a message. The user first creates a file 'message.txt' with the text 'Tamperproof Message: crypto is the coolest!'. Then, they use 'md5sum message.txt' to generate an MD5 hash. Finally, they use 'sha256sum message.txt' to generate a SHA256 hash.

```
maria@ubuntu:~$ echo "Tamperproof Message: crypto is the coolest!" > message.txt
maria@ubuntu:~$ md5sum message.txt
586275d781bbf077bc9b2def6848de9b  message.txt
maria@ubuntu:~$ sha256sum message.txt
8adc78f1bb49c8a86bbff34d9cdebe433217820acbfa186e4556dd69e29872ce  message.txt
maria@ubuntu:~$
```

2.5 Detached Signature

I used Ubuntu to create a detached digital signature and verified it.

First, I constructed a user and id and was prompted to input my email. Then I was able to create a key. After, I created a message to sign. After, I ran a command to verify the signature and make sure it was a good signature. Finally, I confirmed that my signature was good.

```
Activities Terminal Sep 11 11:00
maria@ubuntu: ~
maria@ubuntu: ~ x maria@ubuntu: ~ x maria@ubuntu: ~ x maria@ubuntu: ~ x
maria@ubuntu:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Maria
Email address: mariavalencia@csus.edu
You selected this USER-ID:
"Maria <mariavalencia@csus.edu>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key EDDC76B59891CF22 marked as ultimately trusted
gpg: directory '/home/maria/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/maria/.gnupg/openpgp-revocs.d/5DA10C44C80296C4826F6
ED5EDDC76B59891CF22.rev'
public and secret key created and signed.

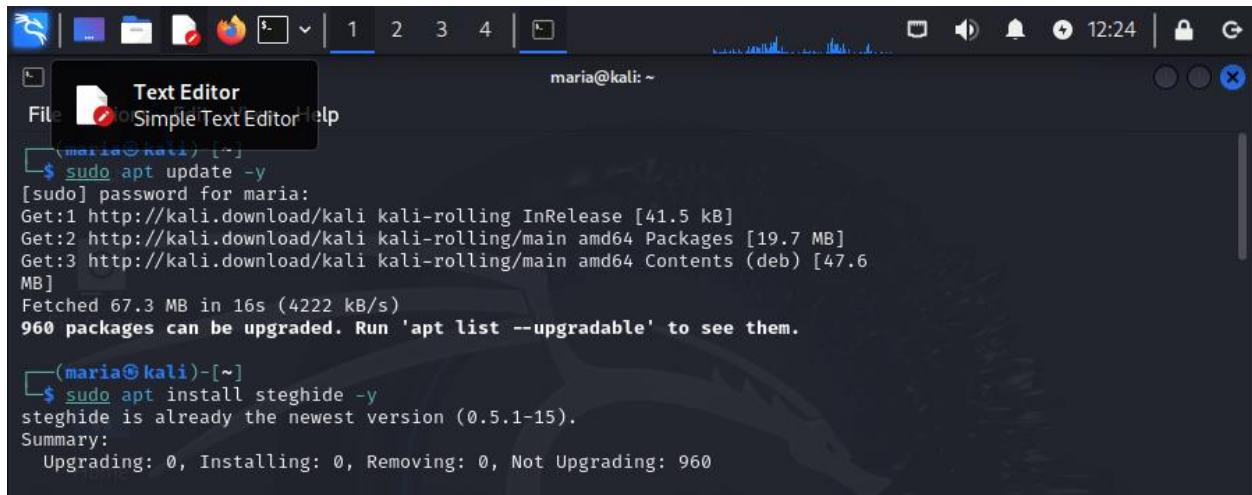
pub rsa3072 2024-09-11 [SC] [expires: 2026-09-11]
SDA10C44C80296C4826F6ED5EDDC76B59891CF22
uid Maria <mariavalencia@csus.edu>
maria@ubuntu:~$ echo "Message integrity and authentication are very cool" > message.txt
maria@ubuntu:~$ gpg --output message.txt.sig --armor --detach-sig message.txt

gpg: can't open 'cat': No such file or directory
gpg: signing failed: No such file or directory
maria@ubuntu:~$
maria@ubuntu:~$
maria@ubuntu:~$
maria@ubuntu:~$
maria@ubuntu:~$
maria@ubuntu:~$
maria@ubuntu:~$ gpg --output message.txt.sig --armor --detach-sig message.txt
maria@ubuntu:~$ cat message.txt.sig
-----BEGIN PGP SIGNATURE-----

iQGzBAABCGAdFiEEXaEMRMgCLsSCb27V7dx2tZiRzyIFAmbh2eMACgkQ7dx2tZiR
zyKR5gwASHfnQri7Y1xastoHe3KGIKX13EYXxN1ip/s/fVSc0CNqZifh9IvMOOS
y8YE8C1ncvCBMG+Z3XN+FWGHQkAU0PVXFxehzY+xOKDZe2s2CgnjXyk3X0oIaZtb
PRqfmoA81SHI+IroJU7cPR4kXXjKzakxrmhhdVgMXZYk0bTE+kQNqJBUA3Ma/yL
FUGkWoNyNC546EQdjh/e1MtuoWq7MoX/jculzEmCKZKZnBZE8181MZhHyvbmUXS
+9Q6xozsL9om5gn0w4nuNMcvbuNVjIsowdrJcaifTySoeh7G/2pL54utfCAycZV
g+uNeZ8HzCkIcETnKUAF3k3lRAEVmwkHTY1plzf6TEmRSL0UEmDfM+zI+kaRoS/
VGb5DqV7Y+EY/gGr5O/SX/dKUHKLa8e4iPPF5XG/FdC41Mfs0nJz3kuBPM+p2m88
LzgSidWfBgEBISu3r7YYZZYzbzf8oi0bL28VGa4prFJnfIprL7nTKwNoLNLrmWJv
1fHotgKY
=v9QZ
-----END PGP SIGNATURE-----
maria@ubuntu:~$ gpg --verify message.txt.sig message.txt
gpg: Signature made Wed 11 Sep 2024 10:56:51 AM PDT
gpg: using RSA key 5DA10C44C80296C4826F6ED5EDDC76B59891CF22
gpg: Good signature from "Maria <mariavalencia@csus.edu>" [ultimate]
maria@ubuntu:~$ gpg --verify message.txt.sig message.txt
gpg: Signature made Wed 11 Sep 2024 10:56:51 AM PDT
gpg: using RSA key 5DA10C44C80296C4826F6ED5EDDC76B59891CF22
gpg: Good signature from "Maria <mariavalencia@csus.edu>" [ultimate]
maria@ubuntu:~$
```

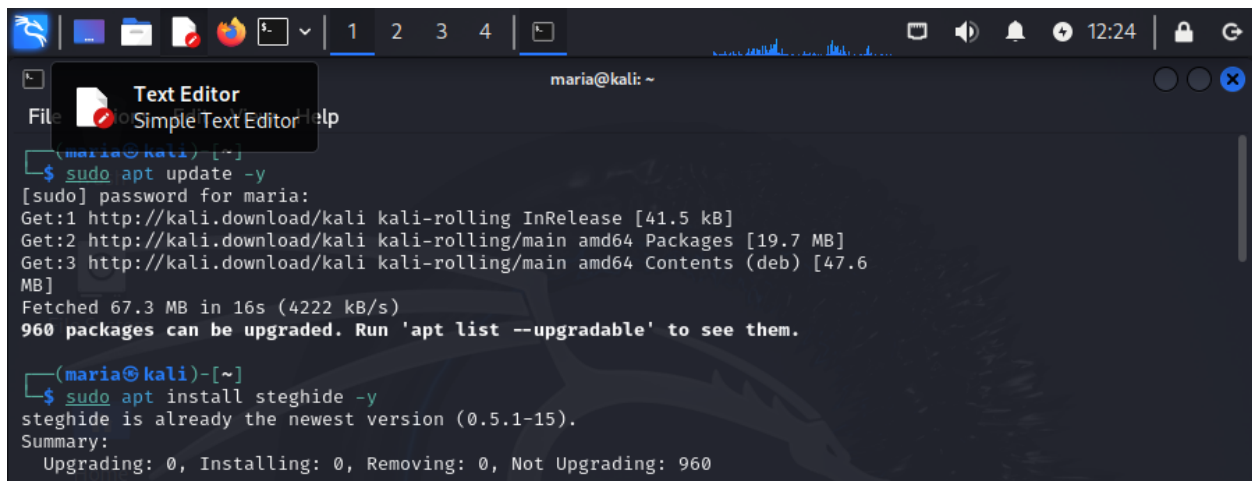
2.6 Steghide

In this exercise I used Kali VM and steganography. After installing steghide, I created a secret message. Then I downloaded an image and embedded it with the secret message. Lastly, I extracted the image with the embedded message and observed its contents.



```
Text Editor
File Edit Simple Text Editor Help
(maria@kali) [~]
$ sudo apt update -y
[sudo] password for maria:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.7 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [47.6 MB]
Fetched 67.3 MB in 16s (4222 kB/s)
960 packages can be upgraded. Run 'apt list --upgradable' to see them.

(maria@kali) [~]
$ sudo apt install steghide -y
steghide is already the newest version (0.5.1-15).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 960
```



```
Text Editor
File Edit Simple Text Editor Help
(maria@kali) [~]
$ sudo apt update -y
[sudo] password for maria:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.7 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [47.6 MB]
Fetched 67.3 MB in 16s (4222 kB/s)
960 packages can be upgraded. Run 'apt list --upgradable' to see them.

(maria@kali) [~]
$ sudo apt install steghide -y
steghide is already the newest version (0.5.1-15).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 960
```



```
Text Editor
File Edit View Help
(maria@kali) [~]
$ ls -la
total 312
drwx----- 16 maria maria 4096 Sep 11 12:18 .
drwxr-xr-x  3 root root 4096 Sep 4 17:43 ..
-rw-----  1 maria maria   0 Sep 4 17:45 .ICEauthority
-rw-----  1 maria maria  49 Sep 11 11:48 .Xauthority
-rw-r--r--  1 maria maria 220 Sep 4 17:43 .bash_logout
-rw-r--r--  1 maria maria 5551 Sep 4 17:43 .bashrc
-rw-r--r--  1 maria maria 3526 Sep 4 17:43 .bashrc.original
drwxrwxr-x 10 maria maria 4096 Sep 11 11:53 .cache
drwxr-xr-x 14 maria maria 4096 Sep 11 12:20 .config
-rw-r--r--  1 maria maria  35 Sep 4 17:45 .dmrc
-rw-r--r--  1 maria maria 11759 Sep 4 17:43 .face
lrwxrwxrwx  1 maria maria   5 Sep 4 17:43 .face.icon -> .face
drwx-----  3 maria maria 4096 Sep 4 17:45 .gnupg
drwxr-xr-x  3 maria maria 4096 Sep 4 17:43 .java
drwxr-xr-x  4 maria maria 4096 Sep 4 17:45 .local
drwx-----  4 maria maria 4096 Sep 11 11:16 .mozilla
-rw-r--r--  1 maria maria  807 Sep 4 17:43 .profile
-rw-r--r--  1 maria maria   0 Sep 4 17:47 .sudo_as_admin_successful
-rw-r-----  1 maria maria   4 Sep 11 11:48 .vboxclient-clipboard-tty7-control.pid
-rw-r-----  4 Sep 11 11:48 .vboxclient-clipboard-tty7-service.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-display-svgx-x11-tty7-control.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-display-svgx-x11-tty7-service.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-draganddrop-tty7-control.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-draganddrop-tty7-service.pid
-rw-r-----  1 maria maria   4 Sep 11 11:48 .vboxclient-hostversion-tty7-control.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-seamless-tty7-control.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-seamless-tty7-service.pid
-rw-r-----  1 maria maria   5 Sep 11 11:48 .vboxclient-vmvga-session-tty7-control.pid
-rw-r----- 11599 Sep 11 12:21 .xsession-errors
-rw-r-----  1 maria maria  8081 Sep 11 11:22 .xsession-errors.old
-rw-r-----  1 maria maria  361 Sep 11 11:22 .zsh_history
-rw-r--r--  1 maria maria 10868 Sep 4 17:43 .zshrc
drwxr-xr-x  2 maria maria 4096 Sep 4 17:45 Desktop
drwxr-xr-x  2 maria maria 4096 Sep 4 17:45 Documents
drwxr-xr-x  2 maria maria 4096 Sep 11 11:53 Downloads
drwxr-xr-x  2 maria maria 4096 Sep 11 11:53 Music
drwxr-xr-x  2 maria maria 4096 Sep 11 11:53 Pictures
drwxr-xr-x  2 maria maria 4096 Sep 4 17:45 Public
drwxr-xr-x  2 maria maria 4096 Sep 4 17:45 Templates
drwxr-xr-x  2 maria maria 4096 Sep 4 17:45 Videos
-rw-rw-r--  1 maria maria 129506 Sep 11 12:19 chop.jpeg
-rw-rw-r--  1 maria maria   13 Sep 11 12:15 secret.txt
(maria@kali) [~]
$ cat secret.txt
chopper wins
```

2.7 Known Plaintext Attack

Plaintext: Break my simple encryption

Ciphertext: rOe nx zlfvrcya rlpevcgba

I broke this encryption using the plain text and cipher text. It took me almost an entire day. I started by making sure the total number of cipher characters matched the plain text. Then I tested a simple shift forward and back within the characters but that did not work. I

eventually realized there could be a 'key'. It took me some time to figure out that "break" was the key. I learned that Vigenère cipher uses a key, and I checked the shift count of every letter. Then I stuck with the shift of just the word "Break" and realized it followed the pattern of backwards Shift 1,17,4,0 and 10.