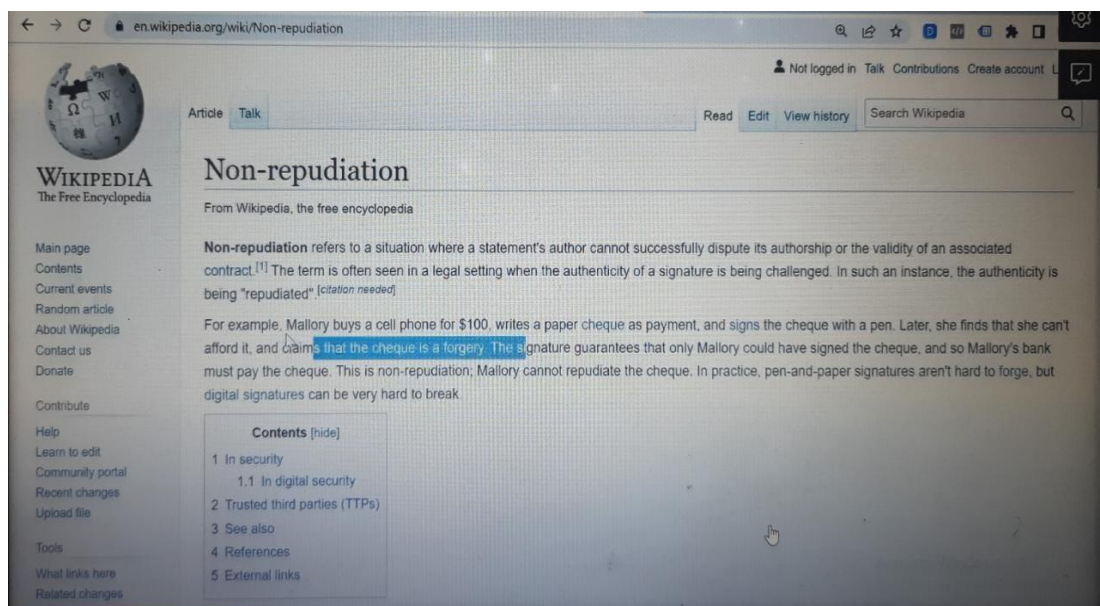


## Cryptography Session No.09

### Summary 29-07-2022

Detailed Discussion on below points –

- In Hybrid Encryption – we have bugs – it can be compromised – to solve this we have to build PKI ( Public Key Infrastructure )
- PKI behind the scene uses Digital Signatures
- Digital Signatures can be used to overcome non repudiation issues
- Brief on Non-repudiation



- To implement Digital Signature we use ECDSA (Elliptic Curve Digital Signature Algorithm)- compared to other algorithms like DSA,ECDSA is faster
- To sign a message that is for authenticity we use Private Key
- To verify the message actually signed by real private key we use Public Key.

- To create Private Key- it is the identity- to sign a message

```
[root@ip-172-31-44-53 ~]# openssl genrsa 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDZjQY63aGYkFd/t2U8CG60fAUZRsCISOqrJJfjtlaO1OYHo5xw
+8k9h0vdbx2bfU+4PaMHPcSPcht3DDecghqGk+7EglRt/E6cOgU8WR6A6o2cd0cK
OoNG+6GV3+ESJ8idwtg/Fm4+v4U+yM7NeGZBJ0nu/NPZwh/zd7ZLSNdP9wIDAQAB
AoGAbuX0xw0z0xHBiNl0QaYKyPZvovLuMor5FUUSKILNHQJqQHqmxPGVJEhT5A1A
ioLWoJTLBmX2TSldltHjh2TLH96rXPwY6oBn+yXZgDe4p/THmoz9HUSIDc+USn2h
V5ObT5P8ePkm253W+yj9oXOpzEyE7mN1OSSP4O55fJPYXAECCQDukavrsfHa+cGS
41t3b98VxU0eQSNL9xrjoOk0/vElJ7Nr2U01+TpuJ3TWCMYBEJchdaaHCl+BQdc
GyVz8MTTAKEA6XI3rUpnVqVR0pdJsyWP0+126gGnp6QP9WlsxoBfYOPVYTPPoHVZ
H+++LYaOC3QkI2Z2ujm80JpJyIW60RuhzQJBAKJ1lsKxccaXr81sgDwMdblom4SP
zQ6NHsWGLWlGd0bbC0bVOswXhV/ABMNbEnrnY52IEsttip/joshpQA79FbsCQQCV
FgqUsxW38jAkGQZKUMy/7cGpxJDLsS0377I2OzmuaRKW1z3cHdVjXfq4nFWO2IdT
zFY6rDD3kAH1x+H9NYZpAkeAhKEPatAXm35JIoJALAwkisfad6xOFdlnfglt0IwD
RwUBM7a1O+fOW7cVix41EQY7sW2G4s40zsQYGmpmqTueKA==
-----END RSA PRIVATE KEY-----
```

- Private key is sensitive , it can be encrypted using AES algorithm and stored in a file- here private key stored in an encrypted way

```
[root@ip-172-31-44-53 ~]# openssl genrsa -aes256 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase:
aborted!
140567935854496:error:0906906F:PEM routines:PEM_ASN1_write_bio:read key:pem_lib.c:385:
[root@ip-172-31-44-53 ~]#
[root@ip-172-31-44-53 ~]# openssl genrsa -aes256 1024 > p.key
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase:
Verifying - Enter pass phrase:
```

```
aws Services Search for services, features, blogs, docs, and more
EC2
Verifying - Enter pass phrase:
[root@ip-172-31-44-53 ~]# cat p.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-256-CBC, 13C4AFF33BD9E4E3E5D91902B1FAB89C

o3NMhP4UPFheBCfhKPK39Qac/iZa89GtKOnqML+OtWcycFY2SP7AF+GXGknWXFhv
89q7awMpufJEFywd/EwBBOT1TFEZqu5Ch7RtRBT6rYEE3Gr1N2A3sowkb0nmDW9k
WaWHFuuVyhrgCazS54dC889p5S2culVXJ9E2iKn51Irxps0ztkI2ZytberyArInt
XnzvaCwtJ/zLoFlhJ3/VPj2S1/MbzrU1KME0UC+mMstd1L+PdC6S0ziAsqtVj5Px
DOTT64y0qUSmaayMcqlrbIZ5x/pvmt0HWw+CE0n3Fi2CVAH5CNPfnpccsy12nM29
04kdVr909hWZavOzYuHOzIBUMIp9Nm7SfisK1r9nKylhMSTL7QMvVolelZLxuJpb
Or9T60niXrMb2O8uHgdPFxOncdib7KhW3wNk8OrMFpHqCH3ITskqfr23vsmA+RNi
lh2LtJhoumI2C6PqQ0G1zHZ3jixf2GkrtFfQPTTRRVMB2nY6COAGN80OPZqDXAAU
2yzqAKd4o0+rydpXSuaTPH+YT9bxeXF650+h1ihJowhAMM2jgFJaUEB/lgxPyCi
dCCwM2N9rb4kMYPMYtg17TerT3jXghibz+qxGUG/r0zaPnKF6uQ5Rx5bR4inJIfG
uXV6EpjsABqGE41SurdGtJfQmA8LPGBBjCG7wvLOGg2fdi7iQGMtoiR9G8jGFGoe
r2F1pt/Q/N738gfHVgfAPNjAanYUgfDfRpcf1ttSwsPI+jURXJXojfVKnJ00/aigh
pJCV7sQ1u5N33F5+Tzm1G1/NXsssGUCRjDzmzTXo+Jp6tAFxiKCCFhQPzQsG6Thu
-----END RSA PRIVATE KEY-----
[root@ip-172-31-44-53 ~]#
```



- Public Key is generated from the Private Key and stored in file

```
MINGW64/c:/Users/Vimal Daga/Documents/cryptography_training_2022
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
$ cd Documents/
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd cryptography_training_2022/
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl genrsa 1024 > private.key
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl rsa -in private.key -pubout > public.key
writing RSA key
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

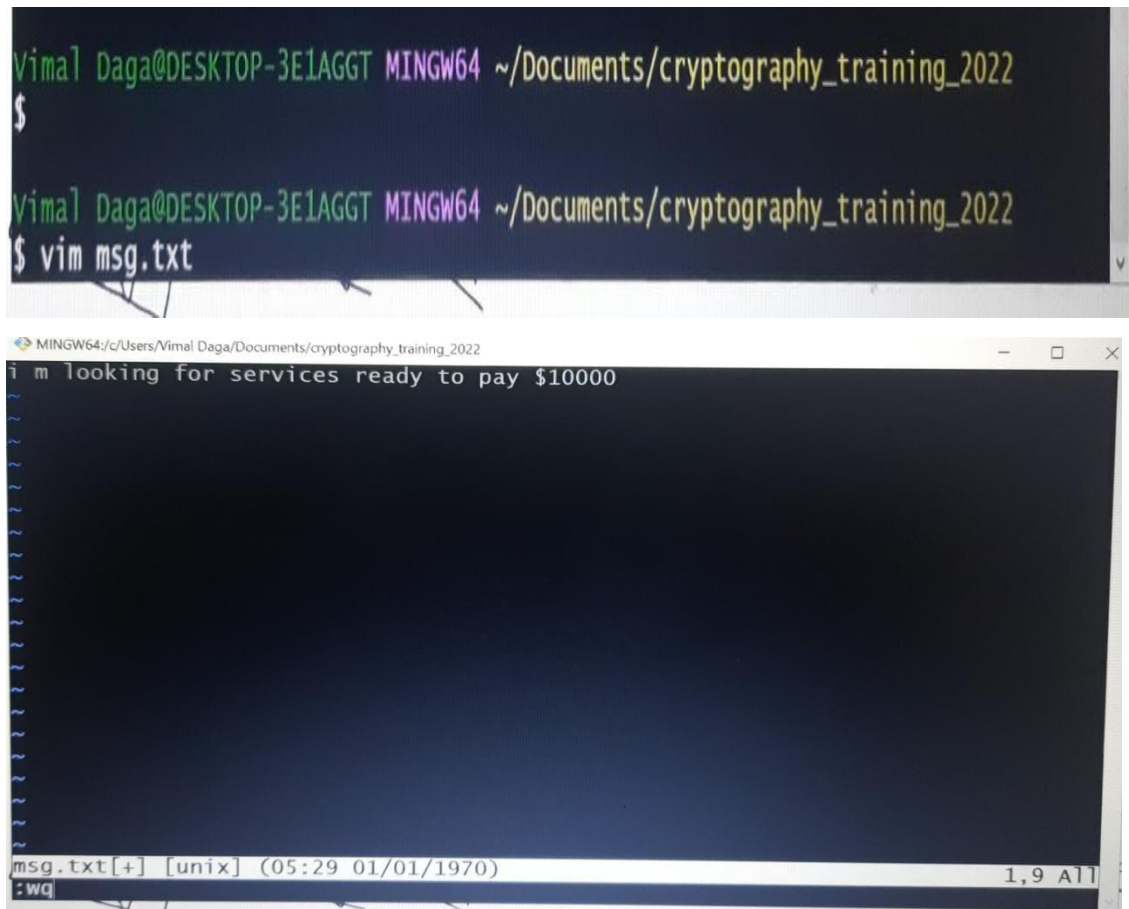
```
MINGW64/c:/Users/Vimal Daga/Documents/cryptography_training_2022
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat public.key
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD1lhjk7MQN3P/lG+CgAm7HKC6b
tvh9BGmA4/urkpdFCG5lVB4lkwT8daM2+Y743j/SrSqzUuUgdgGRU33Xvk6bX1Sx
W4ZYMqrW/25NKzbFTx/TI7mxxrTfGBud42cNYEE+2JM5mTPT1x8bwOivJlpPXXS
undr9ozmcQBQKFnrwIDAQAB
-----END PUBLIC KEY-----
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

- Some of the servers like SSH – for remote login- automatically creates Private and Public Keys

```
EC2
┌───┴───┐
┌───┴───┐ Amazon Linux 2 AMI
└───┴───┘

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-45-91 ~]$ sudo su - root
[root@ip-172-31-45-91 ~]# cd /etc/ssh
-bash: cd: /etc/ssh: No such file or directory
[root@ip-172-31-45-91 ~]# cd /etc/ssh/
[root@ip-172-31-45-91 ssh]# ls
moduli      sshd_config      ssh_host_ecdsa_key.pub  ssh_host_ed25519_key.pub  ssh_host_rsa_key.pub
ssh_config  ssh_host_ecdsa_key  ssh_host_ed25519_key    ssh_host_rsa_key
[root@ip-172-31-45-91 ssh]#
```

- Sign a message for authenticity



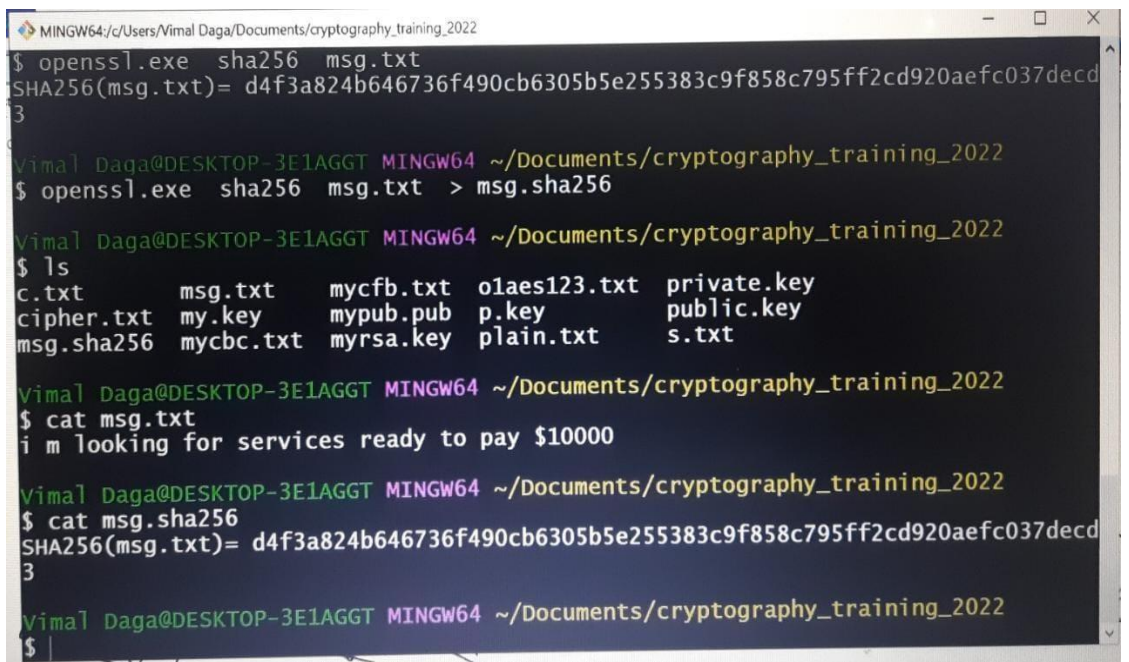
The screenshot shows a terminal window with the following commands and output:

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ vim msg.txt
```

The second part of the screenshot shows the contents of the file `msg.txt` in a text editor:

```
i m looking for services ready to pay $10000
```

- Create a HASH value of the message-to check integrity of data – so that no one can edit the message



The screenshot shows a terminal window with the following commands and output:

```
MINGW64:/c/Users/Vimal Daga/Documents/cryptography_training_2022
$ openssl.exe sha256 msg.txt
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3
3
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl.exe sha256 msg.txt > msg.sha256
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ ls
c.txt      msg.txt    mycfb.txt  o1aes123.txt  private.key
cipher.txt my.key     mypub.pub  p.key         public.key
msg.sha256 mycbc.txt  myrsa.key  plain.txt     s.txt
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.txt
i m looking for services ready to pay $10000
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.sha256
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3
3
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```



- Sign a message- technically we sign the hash – this creates the signature on the data and we can save it in a file

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl.exe rsautl -sign -inkey private.key -in msg.sha256 -out msg.sig

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.txt
i m looking for services ready to pay $10000

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.sha256
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.sig
S3f ( ) ~ wá = wj : d O TV = 2LB s ^ã y
F d SLiu # L

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

- Before the client using the data- they have to check data integrity using hash – to check whether data is edited or changed at the client end.

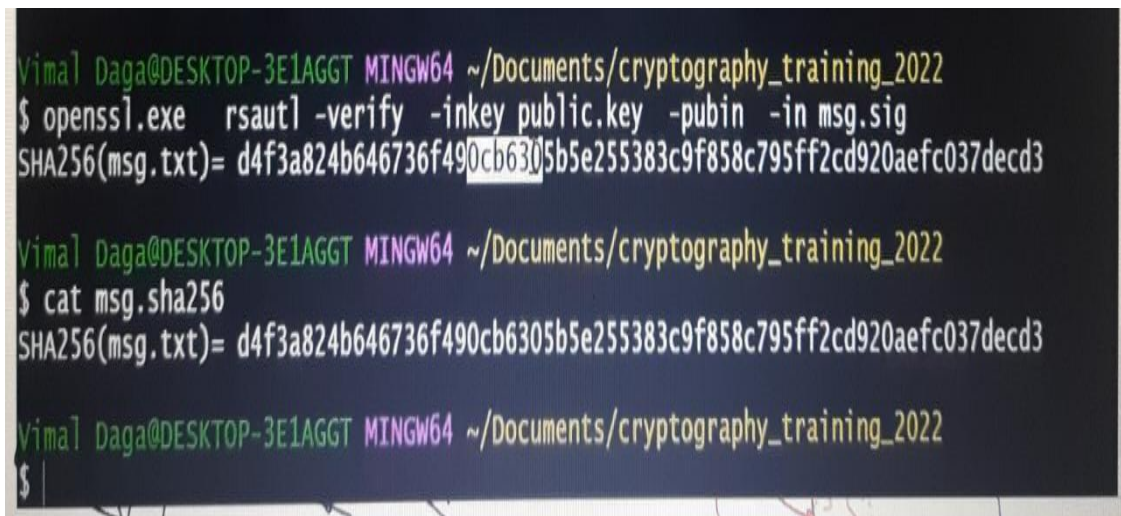
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.txt
i m looking for services ready to pay $10000

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl.exe sha256 msg.txt
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.sha256
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

- To verify the message actually signed by real private key, we use Public Key- to check hash is sent by actual server



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ openssl.exe rsautl -verify -inkey public.key -pubin -in msg.sig
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$ cat msg.sha256
SHA256(msg.txt)= d4f3a824b646736f490cb6305b5e255383c9f858c795ff2cd920aefc037decd3

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/cryptography_training_2022
$
```

- Brief on PKI (Public Key Infrastructure) – how to excel the Public Key for verification of the Signatures.

Important Links –

Hash13 link for Extra Sessions and session recording -

<https://learning.hash13.com/>

Community Link to post Query, Doubts and share your blogs -

<https://hash13-community.circle.so/home>