Exercise 7

Abd El Rahaman Shehata

Exercise 7

1

1.1 - Use the gpg command-line tool to create a new public/private key-pair for SMS. Depending on availability of entropy on your (virtual) system, this may take some time

```
gpg --generate-key

public and secret key created and signed.

pub   ed25519 2022-11-02 [SC] [expires: 2024-11-01]
      678CFF874EE2C6DE41F9DF21E78DE2F174B5EAD4
uid                      SMS <s2010237022@fhooe.at>
sub   cv25519 2022-11-02 [E] [expires: 2024-11-01]
```

```
gpg --armor --export  678CFF874EE2C6DE41F9DF21E78DE2F174B5EAD4

-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEY2JxZxYJKwYBBAHaRw8BAQdA5x2FIfvuX/XNT1hPBlNP3Btsfm7eawY5ZrqL
BdolR3G0GlNNUyA8czIwMTAyMzcwMjJAZmhvb2UuYXQ+iJkEExYKAEEWIQRnjP+H
TuLG3kH53yHnjeLxdLXq1AUCY2JxZwIbAwUJA8JnAAULCQgHAgIiAgYVCgkICwIE
FgIDAQIeBwIXgAAKCRDnjeLxdLXq1DjxAPwKUPe/WRpxsiguHKVexQLGpILip+5U
x0gCYwIhPY5TLAD/ehxB88L+WcQehemfKj4Fk76LK0xns0quAkMyKeFV+gy4OARj
YnFnEgorBgEEAZdVAQUBAQdA8VZIxVTKd3AcUX2bk6cXVIieM959oWJcyUPnZfXX
fVgDAQgHiH4EGBYKACYWIQRnjP+HTuLG3kH53yHnjeLxdLXq1AUCY2JxZwIbDAUJ
A8JnAAAKCRDnjeLxdLXq1PycAQC3REGjp3liz5WBdwTvihqAvnysKi2lMvfh8vw3
t/+jgwEA3sv+Eico5dJnO6ASW79Xr4rLO4MY1KV0pCJGr0QL+AU=
=b9mv
-----END PGP PUBLIC KEY BLOCK-----
```

1.4 - Arbitrarily choose a data file, and cryptographically sign it with your newly created key-pair using gpg

```
gpg -s -r SMS Documentation.md
```

1.5 - Verify the signature on the command-line.

```
gpg --verify -r SMS Documentation.md.gpg

gpg: Signature made Wed  2 Nov 14:35:55 2022 CET
gpg:                using EDDSA key
 ↪  678CFF874EE2C6DE41F9DF21E78DE2F174B5EAD4
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m,
 ↪  0f, 1u
gpg: next trustdb check due at 2024-11-01
gpg: Good signature from "SMS <s2010237022@fhooe.at>" [ultimate]
```

1.6 - Symmetrically encrypt any data file using AES-256.

```
gpg -c --cipher-algo AES256 -r SMS Documentation.md

gpg: WARNING: recipients (-r) given without using public key
 ↪  encryption
File 'Documentation.md.gpg' exists. Overwrite? (y/N) N
Enter new filename: Documentation.md.aes.gpg
```

2

2.1 - Public key authentication: Use the OpenSSH tool `ssh-keygen` to generate a new asymmetric key-pair on the client machine, using the RSA version 2 algorithm. Assure to provide a passphrase for encrypting your private key. Once done and unless defined otherwise, the public key usually resides at `~/.ssh/id_rsa.pub`, while the private key is located at `~/.ssh/id_rsa`. Copy the public key to the server machine, and append its contents to the `authorized_keys` file on the server. Log in from client to server using public-key authentication.

— Describe the process in your protocol.

1. This is explained in the question.
2. I saved it at `~/.ssh/sms`, because I usually have separate keys for different services.
3. Copying the content of `~/.ssh/sms.pub` into `~/.ssh/authorized_keys` on the server. (I used a cloud provider, so I passed the key in the web-app)
4. If I would be running a local server I would have to start the daemon `sshd` and allow ssh through the firewall with `sudo ufw allow ssh`
5. Now connect to the server with `ssh [user@]server_domain_or_ip_address`

— How does public key authentication work?

This method works by sending a signature created with a private

> key of the user. The server MUST check that the key is a valid authenticator for the user, and MUST check that the signature is valid. If both hold, the authentication request MUST be accepted; otherwise, it MUST be rejected.
>
> – RFC4252

— Why is it good practise to encrypt the private key?

Because even if someone gets access to them, they are useless

— What actions have to be taken in order to force your server to only use public-key authentication, and forbid password-based authentication

In the `/etc/ssh/sshd_config` file two lines need to be changed/added

```
PasswordAuthentication no
PubkeyAuthentication yes
```

2.2 Proxying: Create a transparent proxy using the `-D` option. Edit your Firefox' network configuration to use the specified port at your machine as SOCKS proxy. When surfing the net, Firefox will encrypt your data, send it to the SSH server, which in turn decrypts it and actually performs the HTTP(S) requests. Note: If you have access to a remote online SSH server, you can easily circumvent LAN firewall restrictions by binding the remote SSH server to any allowed port (e.g. 80, 443, etc.).

1. Start SSH connection to server

   ```sh
   ssh -D 8080 abdo@68.183.218.196 -i ~/.ssh/sms
   ```

2. Change OS Settings: SOCKS Proxy - localhost:8080
3. Start surfing `sh curl --socks5 localhost:8080 https://www.google.com`
    1. If the first step is skipped curl throws the following Error `curl: (97) Received invalid version in initial SOCKS5 response.`

2.3 - Tunneling: Create an encrypted TCP tunnel for arbitrary data transmissions using the `-L` option. On your client, bind port 1234 as local tunnel endpoint which routes incoming data to the server. Force the server to route incoming connections to a particular service (e.g. your e-mail IMAP server) and its corresponding port (e.g. 143). Try your secure connection by accessing localhost:1234, e.g. via Netcat. Provide a screenshot of the working tunnel access.

- Setup: On the server I used Dovecot
- Client Connections:

```
ssh -L 0.0.0.0:1234:68.183.218.196:143 abdo@68.183.218.196
 ↪  -i ~/.ssh/sms

ncat localhost 1234
* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE
 ↪  IDLE LITERAL+ STARTTLS AUTH=PLAIN] Dovecot (Ubuntu)
 ↪  ready.
```

2.4 - Key visualization: Edit your SSH configuration file to always show visual host keys. Give a reason, why such an option exists. Should it be used?

Source: https://askubuntu.com/a/1030640/1644052

- In the `/etc/ssh/sshd_config` file two lines need to be changed/added `conf VisualHostKey yes`
- It exists because we humans can identify images easier/better than text.
- Yes, it should be used because it can make spotting bad actors easier.

2.5 - SSHFS: Mount a remote directory using `sshfs`. What are the advantages of using SSHFS, opposed to e.g. NFS or SMB (Microsoft local networking protocol) for network file transfers? What's the difference between SSHFS and the `scp` command? Note: You may need to install SSHFS on your machine

- Mounting using sshfs

```
sudo sshfs -o allow_other,default_permissions -o
 ↪  IdentityFile=/home/abdo/.ssh/sms
 ↪  abdo@68.183.218.196:/home/abdo /mnt/droplet

# Output
Enter passphrase for key '/home/abdo/.ssh/sms':

ls -al /mnt/droplet

# Output
drwx------ abdo abdo 4.0 KB Tue Nov  8 13:13:56 2022  .
drwx------ abdo abdo 4.0 KB Tue Nov  8 12:31:06 2022  .cache
drwx------ abdo abdo 4.0 KB Tue Nov  8 12:37:11 2022
 ↪  .config
drwx------ abdo abdo 4.0 KB Tue Nov  8 11:42:40 2022  .ssh
.rw-r--r-- abdo abdo   0 B  Tue Nov  8 13:13:56 2022
 ↪  .sudo_as_admin_successful
drwxr-xr-x root root 4.0 KB Wed Nov  9 12:14:56 2022  ..
```

- Advantages

Every Linux Server has OpenSSH already installed which makes `sshfs` really convenient to use.

- Difference between `scp`

  `scp` only copies files, while sshfs mounts the target directory on the client

  > The scp protocol is outdated, inflexible and not readily fixed. We recommend the use of more modern protocols like sftp and rsync for file transfer instead. According to OpenSSH developers in April 2019, SCP is outdated, inflexible and not readily fixed
  >
  > – OpenSSH

2.6 Reverse tunneling: What is reverse tunneling? Explain and provide an example.

```
ssh -R 4000:localhost:22 abdo@68.183.218.196
```

It's like having a remote desktop application (VNC Viewer) for the terminal (GUI is also possible - X11Forwarding). This is useful because the server IP-Address is fixed and public.

3 - OpenSSL: Assignments

3.1 - Use the `openssl` tool to encrypt any data file of your choice with AES-256, using the OFB block mode, and ensure the result is exported with Base64 encoding (enc option).

- Encryption

```
openssl enc -aes-256-ofb -base64 -in Documentation.md -out
 ↪  Documentation.md.enc

# Output
enter AES-256-OFB encryption password:
Verifying - enter AES-256-OFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

- Decryption

```
openssl enc -d -aes-256-ofb -base64 -in Documentation.md.enc

# Output
enter AES-256-OFB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
---
title: Exercise 7
```

3.2 - Generate a new 2048-bit RSA key pair with OpenSSL. The result is typically stored in a PEM-file - what information does this file contain?

```
# Generate Private Key
openssl genrsa -des3 -out private.pem 2048

# Generate Public Key
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
```

3.3 - Extract the exact cryptographic math parameters of the newly created key pair (modulus, exponents, primes, coefficient).

```
# Private Key
openssl rsa -in private.pem -noout -text

# Public Key
openssl rsa -in public.pem -pubin -noout -text
```

3.4 - Encrypt the private key symmetrically (`enc` option).

```
openssl enc -aes-256-cbc -in private.pem -base64
```

3.5 - Export the public key to a new file.

Already done in 3.2

3.6 - Create a certificate signing request (for a CA) from your previously generated keys.

```
openssl req -new -key private.pem -keyout private.ca.key -out
 ↪  sms.csr

# Output
Enter pass phrase for private.pem:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

3.7 - Use OpenSSL to create a HMAC for the file located at https://delta-xi.net/download/cat.gif, using the SHA-512 hashing algorithm and the key thiscatisnotgrumpy. What's the resulting MAC value?

```
curl https://delta-xi.net/download/cat.gif --output /tmp/cat.gif
 ↪  && openssl dgst -sha512 -hmac "thiscatisnotgrumpy"
 ↪  /tmp/cat.gif

# Output
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time
   ↪  Time  Current
                                 Dload  Upload   Total   Spent
                                  ↪  Left  Speed
100 17201  100 17201    0     0  63596      0 --:--:-- --:--:--
 ↪  --:--:-- 64423

HMAC-SHA2-512(/tmp/cat.gif)=
 ↪  d607ab860ca6d9cb2184af57c3685e55d7addf8aa553c603d2c4e49711ed340b342d9cdf6952d0e82441133c
```

3.8 - Use OpenSSL to generate 16 random bytes, and output the result in Base64 encoding (e.g. useful for automated password generation).

```
openssl rand -base64 16

# Output
RMZvUC1YOOqwhhqGNNq3EA==
```

3.9 - Use OpenSSL to establish a TLS connection to the HTTPS service at delta-xi.net. Which TLS protocol version and cipher is used by default? Deconstruct the cipher-string, and describe its components.

```
openssl s_client delta-xi.net:443

# Output
...
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
...
```

- Protocol version: TLSv1.2
- Cipher: ECDHE-RSA-AES256-GCM-SHA384
  - Key exchange: ECDHE
  - Signatures (authentication): RSA
  - Symmetric encryption: AES256
  - Block mode: GCM
  - Hash algorithm for integrity/MAC: SHA384

3.10 - How can OpenSSL deal with Certificate Revocation Lists, which keep track of invalidated certificates (e.g. stolen private key)?

Source

With the following command: `openssl ca -gencrl -keyfile filename -cert file -out file -config filename` a CRL will be created that OpenSSL can use. After revoking a certificate the list needs to be regenerated to stay active.

4 - Minisign: A small, efficient tool for creating and verifying signatures

4.1 - Create a new keypair

```
minisign -G

# Output
Please enter a password to protect the secret key.

Password:
Password (one more time):
Deriving a key from the password in order to encrypt the secret
 ↪  key... done

The secret key was saved as /Users/abdo/.minisign/minisign.key -
 ↪  Keep it secret!
The public key was saved as minisign.pub - That one can be
 ↪  public.

Files signed using this key pair can be verified with the
 ↪  following command:

minisign -Vm <file> -P
 ↪  RWRJF98OoREoa2bNEezI/nQpC7kwVMmlPQUgm4Fxg0dLaUEwiEGxKsKS
```

4.2 - Sign any file on your system and verify its signature

- Sign

```
minisign -Sm Documentation.md -P
 ↪  RWRJF98OoREoa2bNEezI/nQpC7kwVMmlPQUgm4Fxg0dLaUEwiEGxKsKS

# Output
Password:
Deriving a key from the password and decrypting the secret key...
 ↪  done
```

- Verify

```
minisign -Vm Documentation.md -P
 ↪  RWRJF98OoREoa2bNEezI/nQpC7kwVMmlPQUgm4Fxg0dLaUEwiEGxKsKS

# Output
Signature and comment signature verified
Trusted comment: timestamp:1667399314   file:Documentation.md
 ↪  hashed
```

5 - Setting up a TLS web-server

The setup mixes global and user specific config directories.

The web server used: **NGINX**

```
sudo certbot certonly -v --webroot --agree-tos -w
 ↪  /var/www/letsencrypt -d s2010237022.sytes.net

# Output
"Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator webroot, Installer None
Requesting a certificate for s2010237022.sytes.net
Performing the following challenges:
http-01 challenge for s2010237022.sytes.net
Using the webroot path /var/www/letsencrypt for all unmatched
 ↪  domains.
Waiting for verification...
Cleaning up challenges

Successfully received certificate.
Certificate is saved at:
 ↪  /etc/letsencrypt/live/s2010237022.sytes.net/fullchain.pem
Key is saved at:
 ↪  /etc/letsencrypt/live/s2010237022.sytes.net/privkey.pem
This certificate expires on 2023-02-01.
These files will be updated when the certificate renews.

NEXT STEPS:
- The certificate will need to be renewed before it expires.
 ↪  Certbot can automatically renew the certificate in the
 ↪  background, but you may need to take steps to enable that
 ↪  functionality. See https://certbot.org/renewal-setup for
 ↪  instructions.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ↪  - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:
 ↪  https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ↪  - - - - - - -"

sudo certbot -v --nginx --agree-tos -w /var/www/letsencrypt -d
 ↪  s2010237022.sytes.net
```

```
# Output
"Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Requesting a certificate for s2010237022.sytes.net

Successfully received certificate.
Certificate is saved at:
 ↪  /etc/letsencrypt/live/s2010237022.sytes.net/fullchain.pem
Key is saved at:
 ↪  /etc/letsencrypt/live/s2010237022.sytes.net/privkey.pem
This certificate expires on 2023-02-01.
These files will be updated when the certificate renews.

Deploying certificate
Deploying Certificate to VirtualHost
 ↪  /usr/local/etc/nginx/sites-enabled/s2010237022.conf
Successfully deployed certificate for s2010237022.sytes.net to
 ↪  /usr/local/etc/nginx/sites-enabled/s2010237022.conf
No matching insecure server blocks listening on port 80 found.
Congratulations! You have successfully enabled HTTPS on
 ↪  https://s2010237022.sytes.net

NEXT STEPS:
- The certificate will need to be renewed before it expires.
 ↪  Certbot can automatically renew the certificate in the
 ↪  background, but you may need to take steps to enable that
 ↪  functionality. See https://certbot.org/renewal-setup for
 ↪  instructions.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ↪  - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:
 ↪  https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ↪  - - - - - - -"
```