# HÜ4 - SSH

## 0. Intro

**Used equipment:**

- Ubuntu 24.04 LTS local machine
- Ubuntu 24.04 LTS remote machine
- Chrome Browser

## 1. SSH tunneling

In the first step a pair of ne SSH keys needs to be generated, unless you want to use already existent ones.

```
❯ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/philip/.ssh/id_ed25519):
cryptmeth
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in cryptmeth
Your public key has been saved in cryptmeth.pub
The key fingerprint is:
SHA256:wPQ60jzvVg9qwB/8zDtoxA8nPeTVsJ77u1nSPJEkNzk philip@framework
The key's randomart image is:
+--[ED25519 256]--+
|      .          |
|     o .    .   .|
|      o .    = E |
|     o o  . o = +|
|    ..*oS+ o . o |
|     .o+B B o  o.|
|       +.& + ...+|
|       .B * o  +.|
|       +. .o .=o |
+----[SHA256]-----+
```

For the purpose of this exercise the SSH keys were saved in the `cryptmeth` (private key) file and the `kryptmeth.pub` (public key) file. For this exercise the keys are not protected with a passphrase.

The public fingerprint of the SSH key was installed to the remote host on creation, if that is not possible the fingerprint can be added after remote machine setup with the following command (keep in mind for the second option the right permissions must be set on the remote machine):

```
❯ ssh-copy-id -i cryptmeth.pub root@64.225.100.185
```

After setting up the SSH keys we can check the SSH connection:

```
❯ ssh -i cryptmeth root@64.225.100.185
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jan 19 18:06:35 UTC 2025

  System load:  0.01              Processes:             99
  Usage of /:   20.7% of 8.65GB   Users logged in:       0
  Memory usage: 38%               IPv4 address for eth0: 64.225.100.185
  Swap usage:   0%                IPv4 address for eth0: 10.19.0.6

Expanded Security Maintenance for Applications is not enabled.

7 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


root@cryptographic-mehtods:~#
```

> [NOTE] In this example we used a ssh key which was configured via the web interface of our hosting provider. With cloud providers this is one of the more common options. If ypu configure a remote system yourself keep in mind you will need to enable ssh and setup an inital password access and configure ssh key access after that.

> [WARNING] In this example the `root` account is used. This is NOT recommended for production environments. In production use dedicated accounts with a correct permission setup.

## Forwarding browser traffic

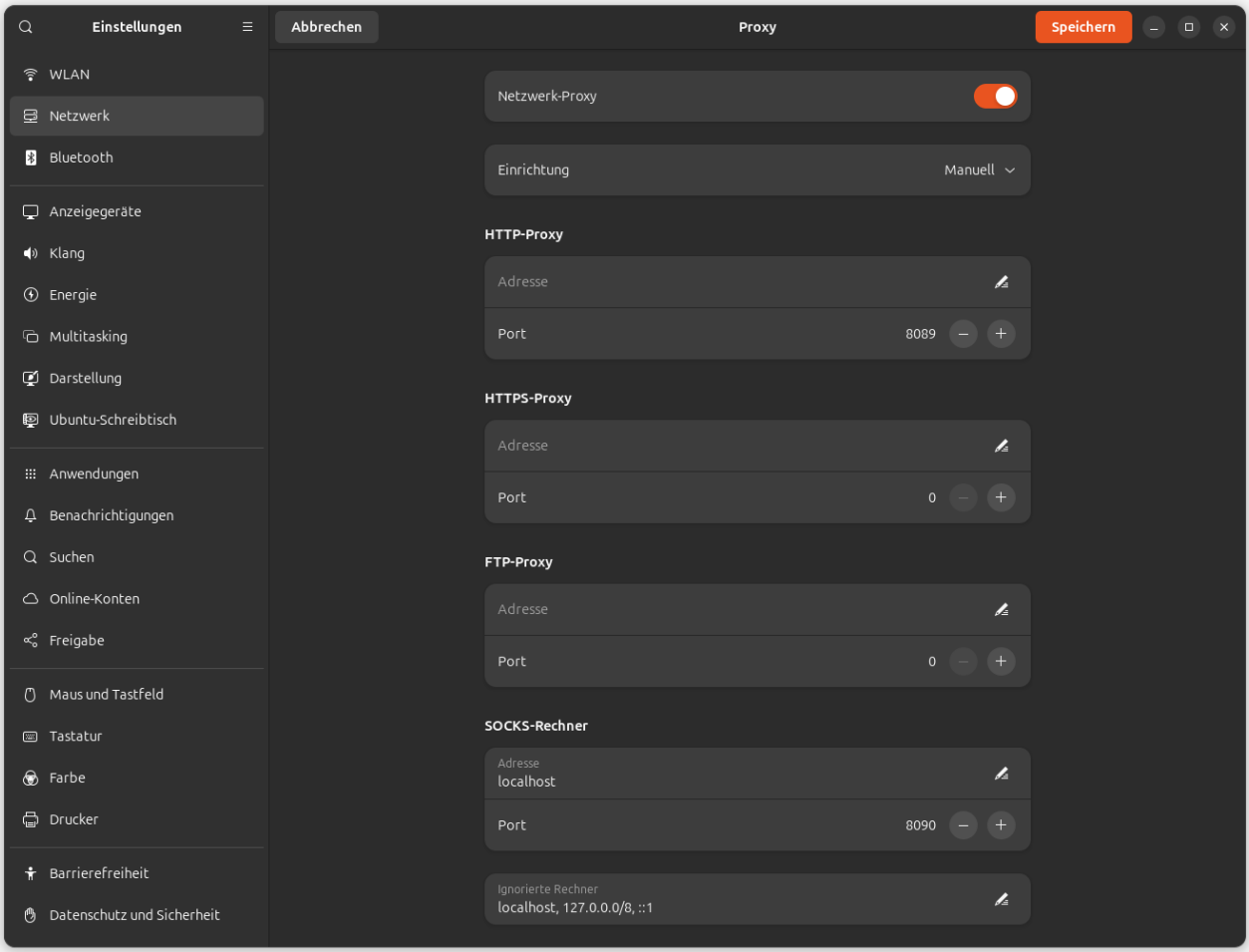
With the following command we create a SOCKS proxy on port `8090`:

```
❯ ssh -i cryptmeth -D 8090 root@64.225.100.185
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jan 19 18:19:37 UTC 2025

  System load:  0.08              Processes:             104
```

```
  Usage of /:   20.7% of 8.65GB   Users logged in:       0
  Memory usage: 37%               IPv4 address for eth0: 64.225.100.185
  Swap usage:   0%                IPv4 address for eth0: 10.19.0.6

Expanded Security Maintenance for Applications is not enabled.

7 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Jan 19 18:06:35 2025 from 62.178.13.69
root@cryptographic-mehtods:~#
```

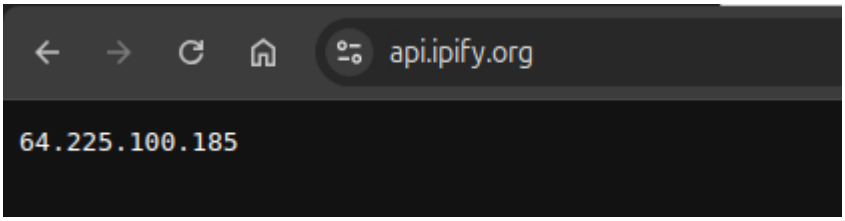
Note that the SSH session starts normally and we can use the session to start/stop tasks on the remote machine. The session can also be stopped normally.

In the `network settings` of our operating system we can now conmfigure our SOCKS proxy to be used to tunnel all network traffic through our remote machine.

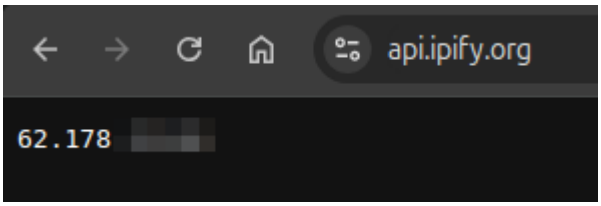


For the host we use our localhost and the tunneled port `8090`. After saving the settings we can check if our proxy is working.

First we check our IP address with our proxy active:




Then we compare this to our IP address without our proxy active:




> [NOTE] The second IP was partially blurred for privacy reasons.

When using the verbose mode whe can also see connections which are tunneled via the SSH tunnel proxy:

```
root@cryptographic-mehtods:~# debug1: Connection to port 8090 forwarding to
socks port 0 requested.
debug1: channel 3: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 4: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 5: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 6: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 7: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 8: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 9: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 10: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 11: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 12: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 13: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 14: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 15: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 16: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
```

```
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 17: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 18: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 19: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 20: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
debug1: Connection to port 8090 forwarding to socks port 0 requested.
debug1: channel 21: new dynamic-tcpip [dynamic-tcpip] (inactive timeout: 0)
```

## SSH audit

> [NOTE] for the purpose of this exercise we will use a SSH server that is already installed and running. This would be the case in most linux distributions anyway.

If you need to install a SSH server use the following steps on a ubuntu system:

```
sudo apt install openssh-server -y
```

After the installation enable the ssh service by using:

```
sudo systemctl enable ssh
```

To start the ssh server immediately and not wait for a full system restart use:

```
sudo systemctl start ssh
```

After that check if the ssh service is up and running with:

```
sudo systemctl status ssh
```

Using the connection setup from part one of this exercise we connect to our remote machine:

```
❯ ssh -i cryptmeth root@64.225.100.185
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)
[...]
root@cryptographic-mehtods:~#
```

Now we can install the `ssh-audit`tool via pip on our remote machine:

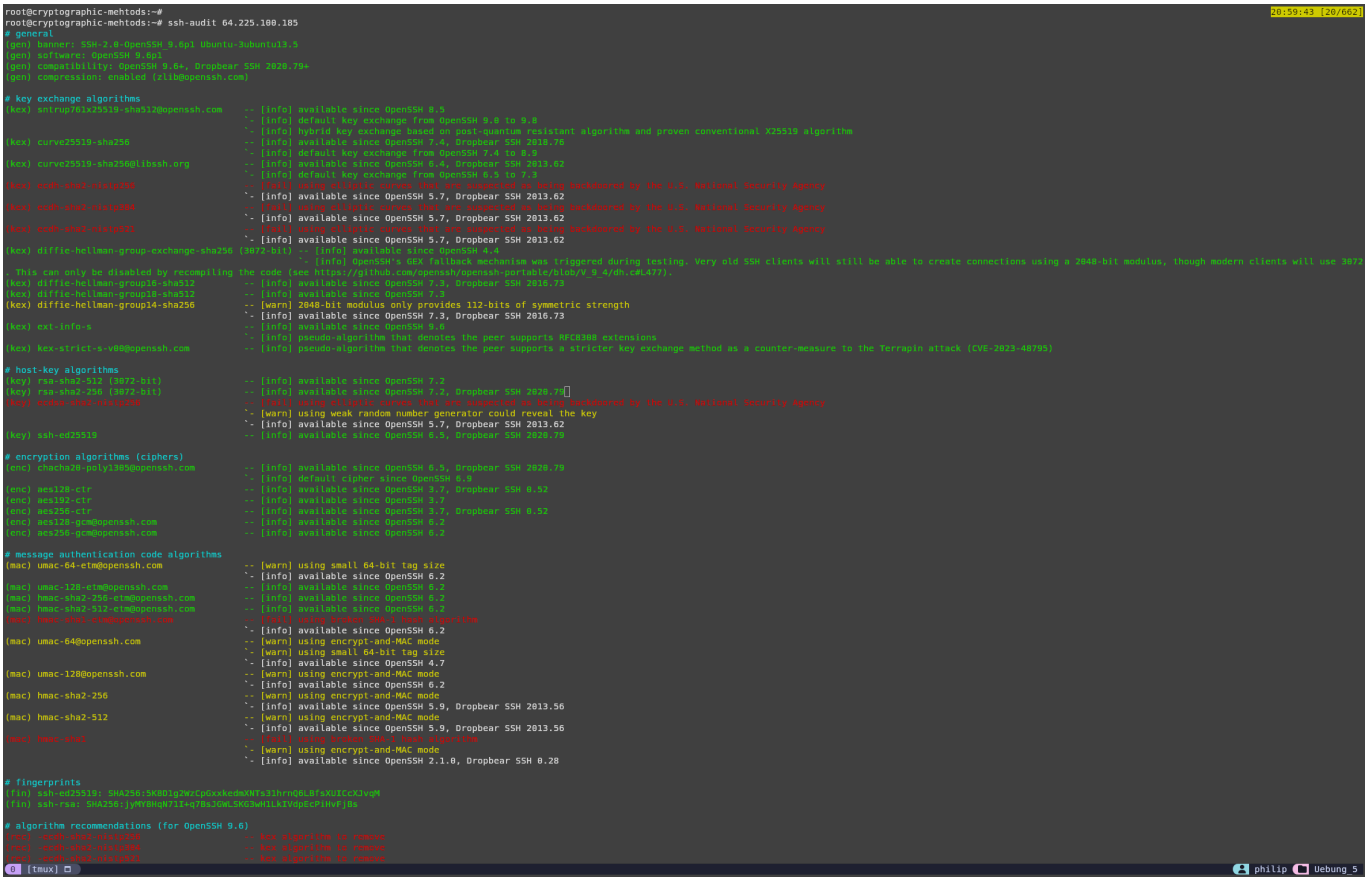
```
root@cryptographic-mehtods:~# sudo apt install pipx -y
[...]

root@cryptographic-mehtods:~# pipx install ssh-audit
  installed package ssh-audit 3.3.0, installed using Python 3.12.3
  These apps are now globally available
    - ssh-audit
⚠   Note: '/root/.local/bin' is not on your PATH environment variable.
These apps will not be globally accessible until your PATH is updated. Run
`pipx ensurepath` to automatically add
    it, or manually modify your PATH in your shell's config file (i.e.
~/.bashrc).
done! ✨ 🌟 ✨
```

First we install `pipx` which is the recommended way of installing Python packages globally. This is done to not interfere with the standard Python installation on your system. `pipx` will then manage the Python-venvs for your installed packages.

After we install `ssh-audit` we can check the configuration of our remote machine.

```
root@cryptographic-mehtods:~# ssh-audit 64.225.100.185
```

The following output was generated by ssh-audit:



As you can see in the output we get a variety of recommended cipher-suites which should be removed from our SSH servers config in order to harden it against attacks.

## Editing the SSH server config

In order to change the cipher suites which will be advertised by our server we need to edit the `/etc/ssh/sshd_config.d`. The changes we apply will be used to harden our server configuration. To get a configuration as clean as possible we will save our changes in a new file.

The new `/etc/ssh/sshd_config.d/hard.conf` will be loaded by the SSH server and overwrite the default conf because it is placed in the `sshd_config.d` folder.

> [WARNING] Proceed with caution. Misconfiguration might result in lost access to your server. Configure your ssh server while keeping a seperate connection open in case you need emergency access. Leave the emergency session only when you are really sure that your configuration did not break your access.

With our new hardend config file:

```
# Only allow recommended kex algorithms
KexAlgorithms  sntrup761x25519-sha512@openssh.com,curve25519-
sha256,curve25519-sha256@libssh.org,diffie-hellman-group-exchange-
sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512
# Host key algorithms - allowed for host keys
HostKeyAlgorithms rsa-sha2-512,rsa-sha2-256,ssh-ed25519
# Message authentication code algorithms
MACs umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-
etm@openssh.com
```

We only allow the ssh-audit recommended KEX, MAC and host key algorithms. With the following command we restart our SSH server.

```
root@cryptographic-mehtods:~# sudo systemctl restart ssh
```

With `ssh-audit` we can now see that only recommended cipher-suites are active.


To manually check the advertised cipher-suites we can start a ssh-connection with `-vvv`.

```
[...]
debug2: peer server KEXINIT proposal
debug2: KEX algorithms: sntrup761x25519-sha512@openssh.com,curve25519-
sha256,curve25519-sha256@libssh.org,diffie-hellman-group-exchange-
sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512,ext-
info-s,kex-strict-s-v00@openssh.com
debug2: host key algorithms: rsa-sha2-512,rsa-sha2-256,ssh-ed25519
debug2: ciphers ctos: chacha20-poly1305@openssh.com,aes128-ctr,aes192-
ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
debug2: ciphers stoc: chacha20-poly1305@openssh.com,aes128-ctr,aes192-
ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
debug2: MACs ctos: umac-128-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,hmac-sha2-512-etm@openssh.com
debug2: MACs stoc: umac-128-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,hmac-sha2-512-etm@openssh.com
[...]
```

Here we can see that only the configured suites are advertised by the SSH server.

## Removing unused SSH keys

It is a best practice to remove unused SSH keys from your servers to further harden them. To remove a SSH key you need to remove the corresponding line in the `~/.ssh/authorized_keys` file. After removal the key would no longer be allowed to authenticated for connections to the server.