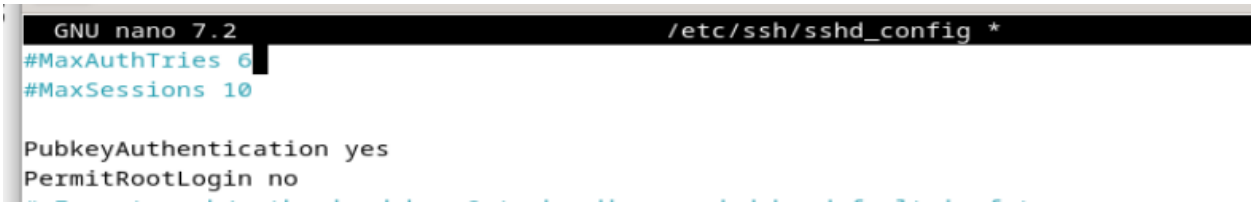
Summary of Deployment

Server	Purpose	Linux Distribution
S1	Edge Server	**Debian**
S2	SSH Server	**Ubuntu**
S3	Web Server (HTTPS)	**Linux Mint**
S4	Secure IP Payloads (IPsec)	**Kali Linux**

I have installed 4 operating systems for the Assignment. Details of which are in above table.



Public Key Authentication is kept on while the Root Login is Disabled in Server 1 and Server 2.

Apache2 server is installed as webserver. In Server 3

```
mint@mint-m:~$ sudo apt install apache2 -y
[sudo] password for mint:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Strongswan is Installed in Server 4 for secure IP payloads

Task A: Deploy S1 (Edge Server)

Solution Description:

The edge server (S1) was configured to ensure secure access using the following cryptographic mechanisms:

Public-Key Cryptography: Public-key authentication was implemented to eliminate password-based access, reducing the risk of brute-force attacks.

No-Key/No-Login Policy: Only users with valid private keys can log in. Unauthorized access attempts fail without an associated key.

Disabled Root Login: This prevents direct root access, further securing the server.

Implementation Methodology:

Configured OpenSSH Server:

(a) Installed OpenSSH **using sudo apt install openssh-server**.
(b) Edited /etc/ssh/sshd_config to:
1. Disable root login: PermitRootLogin no.
2. Disable password authentication: PasswordAuthentication no.
3. Enable public-key authentication: PubkeyAuthentication yes.

(c) Restarted the SSH service: **sudo systemctl restart ssh**.

Generated SSH Key Pair:

On the admin system, created a key pair with ssh-keygen.

Transferred the public key to S1 using ssh-copy-id.

Tested Access:

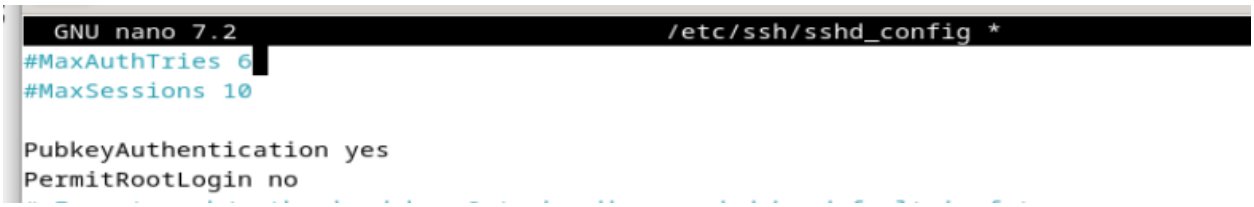
- Successfully logged in using the private key.
- Verified password-based login was rejected.

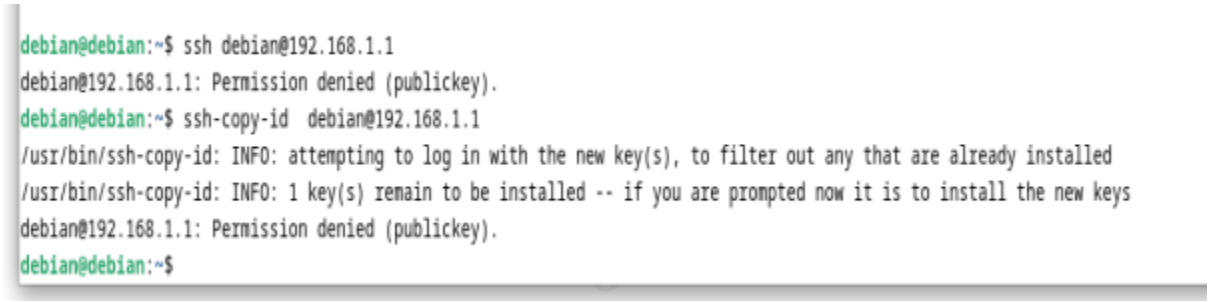
Testing Methodology:

- Attempted to log in without a key: Rejected.
- Logged in with the correct private key: Successful.

Summary:

This configuration ensures secure, controlled access to the edge server by utilizing public-key cryptography and enforcing a no-password policy.





Task B: Deploy S2 (Dedicated SSH Server)

Solution Description:

The dedicated SSH server (S2) was configured to securely manage SSH communications between S1 and S3 using:

- **Public-Key Cryptography:** All SSH connections are authenticated using public keys, avoiding password usage.
- **No-Key/No-Login Policy:** Users without authorized keys are denied access.
- **Disabled Root Login:** Enhances security by disallowing root-level access.

Implementation Methodology:

Configured OpenSSH:

- Installed and configured OpenSSH as done for S1.
- Exchanged public keys between S1, S2, and S3 using ssh-copy-id.

Allowed Communication Between Servers:

- Added public keys of S1 and S3 to /home/username/.ssh/authorized_keys on S2.
- Ensured S2's public key was authorized on S1 and S3.

Tested Connectivity:

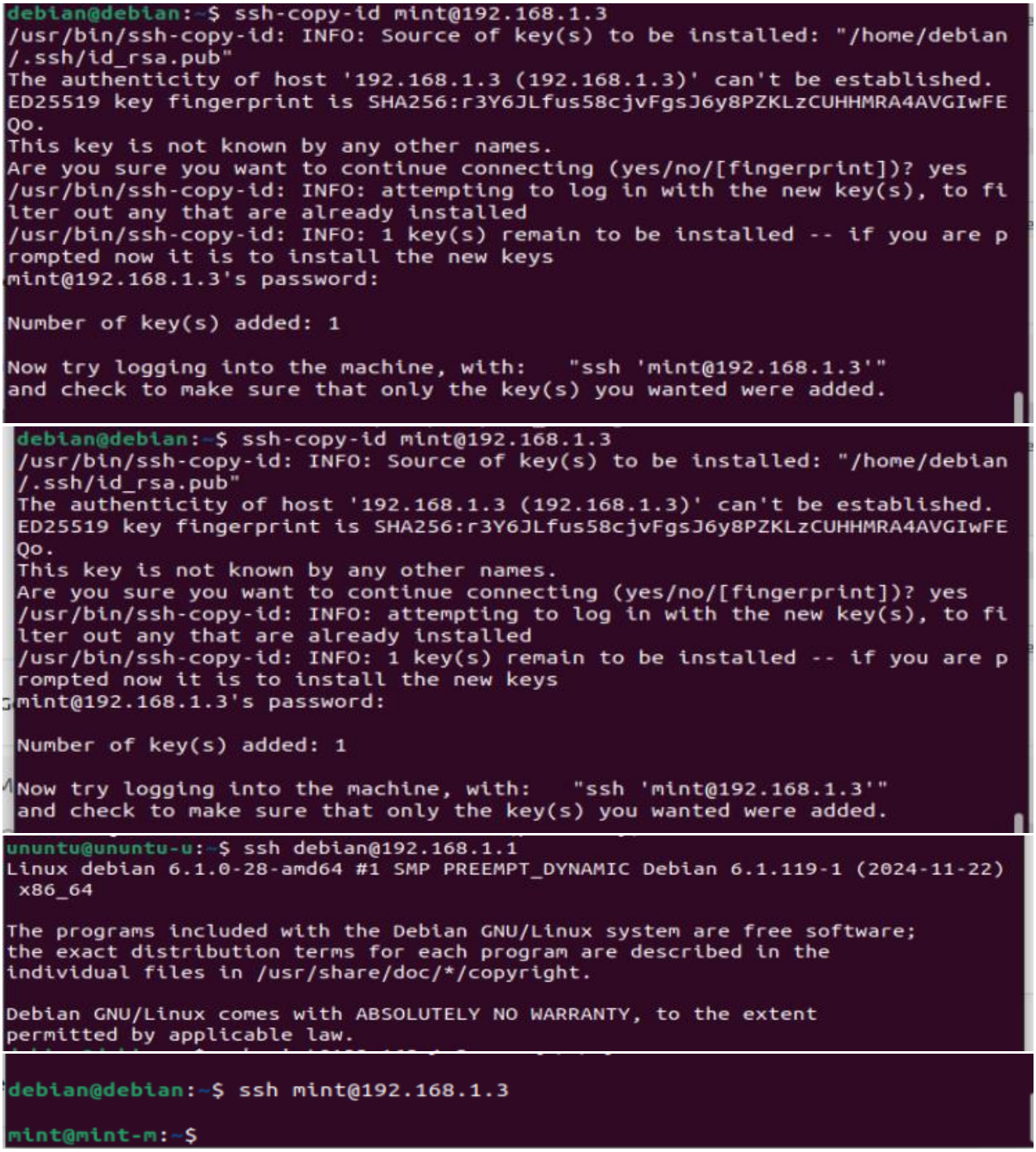
- SSH login from S1 to S2 and S3 to S2 was tested successfully using public-key authentication.

Testing Methodology:

- Verified key-based login between S1, S2, and S3.
- Verified rejection of unauthorized login attempts.

Summary:

This setup ensures a secure channel for SSH communication between the servers, using public-key cryptography to authenticate connections.



S2 can successfully connect with S1 and S3

Task C: Deploy S3 (Web Server)

Solution Description:

The web server (S3) was configured to securely host web applications using:

- **HTTPS Encryption:** Enabled SSL/TLS to encrypt communication between clients and the server.
- **Digital Certificates:** Created and installed a self-signed certificate to establish trust and secure web traffic.

Implementation Methodology:

1. **Installed and Configured Apache:**
    - Installed Apache: sudo apt install apache2.
    - Enabled SSL module: sudo a2enmod ssl.
    - Restarted Apache: sudo systemctl restart apache2.
2. **Generated a Self-Signed Certificate:**
    - Created a private key and Certificate Signing Request (CSR):

      sudo openssl req -new -newkey rsa:2048 -nodes -keyout /etc/ssl/private/s3.key -out /etc/ssl/private/s3.csr

    - Signed the CSR to create a self-signed certificate:

      sudo openssl x509 -req -days 365 -in /etc/ssl/private/s3.csr -signkey /etc/ssl/private/s3.key -out /etc/ssl/private/s3.crt

Configured HTTPS in Apache:

1. Edited **/etc/apache2/sites-available/default-ssl.conf** to point to the certificate and key:

   SSLCertificateFile /etc/ssl/private/s3.crt
   SSLCertificateKeyFile /etc/ssl/private/s3.key
2. Enabled the SSL site:
   sudo a2ensite default-ssl
   sudo systemctl reload apache2

Tested HTTPS Access:

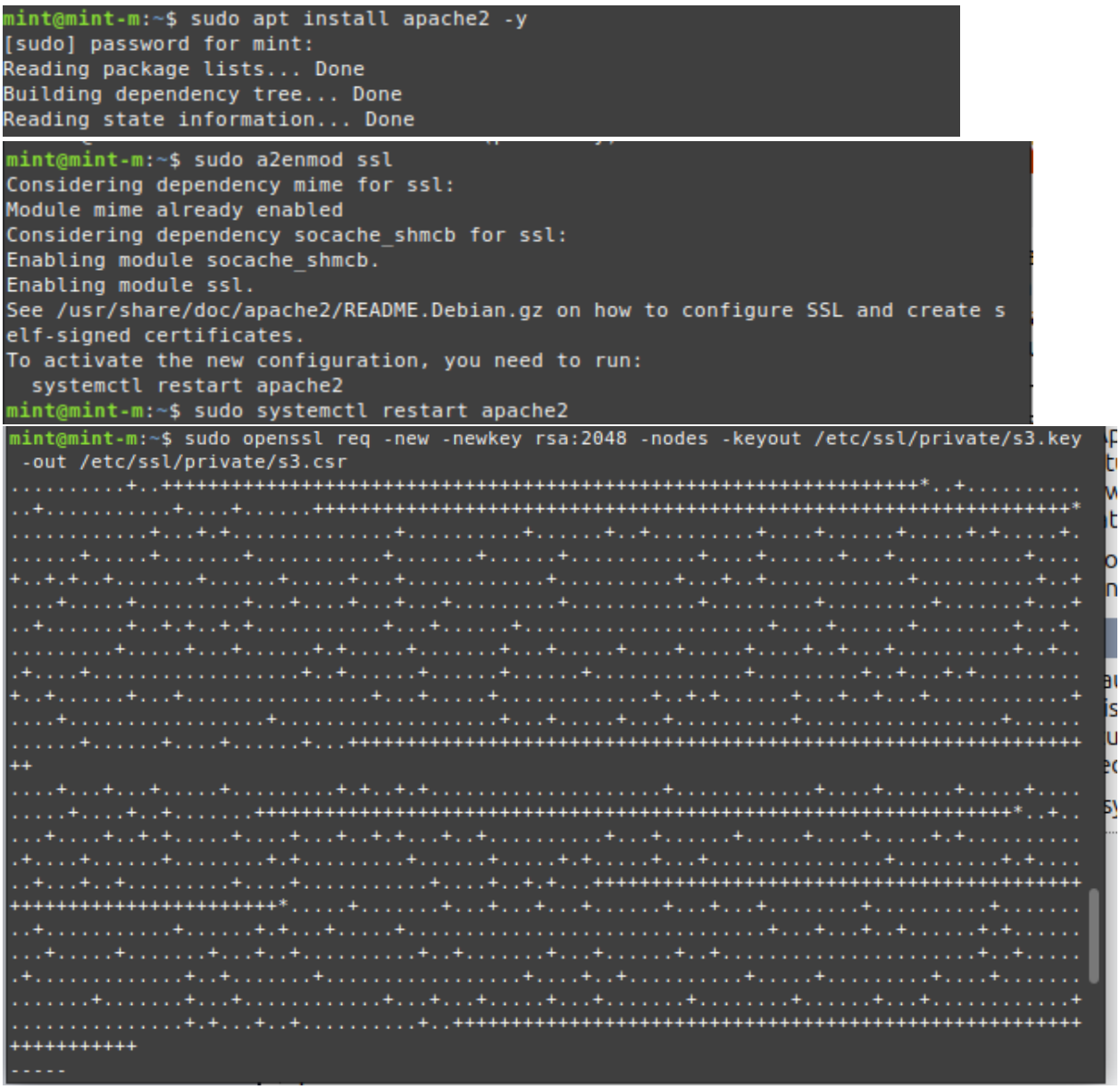
- Accessed the web server using https://192.168.1.3.
- Verified the certificate in the browser.

Testing Methodology:

- Confirmed the HTTPS connection in a browser.
- Verified the certificate was being used.

Summary:

The web server is now secured with HTTPS, ensuring all communications are encrypted. While a self-signed certificate was used, it can be replaced with a CA-signed certificate for production use



```
mint@mint-m:~$ sudo openssl x509 -req -days 365 -in /etc/ssl/private/s3.csr -signkey /etc/ss
l/private/s3.key -out /etc/ssl/private/s3.crt
Certificate request self-signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
mint@mint-m:~$ sudo nano /etc/apache2/sites-available/deault-ssl.conf
mint@mint-m:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
mint@mint-m:~$ sudo systemctl reload apache2
```


```
mint@mint-m:~$ sudo openssl x509 -req -days 365 -in /etc/ssl/private/s3.csr -signkey /etc/ss
l/private/s3.key -out /etc/ssl/private/s3.crt
Certificate request self-signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
```


Ubuntu

Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

The Apache Server is Turned On Successfully

Task D: Public Key Infrastructure (PKI)

a Purpose or Need for PKI

Public Key Infrastructure (PKI) provides a framework for secure communication by:

- **Authenticating entities:** Ensures that the communicating parties are legitimate by verifying digital certificates.
- **Data Integrity:** Guarantees that the transmitted data is not altered during transit.
- **Confidentiality:** Encrypts data to prevent unauthorized access.
- **Trust Establishment:** Facilitates trust through certificates issued by trusted Certificate Authorities (CAs).

b. Components or Participants in PKI

1. **Certificate Authority (CA):**
   - Issues, manages, and revokes digital certificates.
2. **Registration Authority (RA):**
   - Verifies the identity of entities requesting certificates and forwards requests to the CA.
3. **Certificate Repository:**
   - A centralized location for storing issued certificates and Certificate Revocation Lists (CRLs).
4. **End Entities:**
   - Devices or users that utilize digital certificates for secure communication.

End Entity ---> Registration Authority ---> Certificate Authority ---> Certificate Repository

c. Types of Cryptographic Systems Used in PKI

1. **Asymmetric Cryptography:**
   - Used for encryption and digital signatures.
   - Example: RSA (Rivest-Shamir-Adleman) public-private key pairs.
2. **Hashing Algorithms:**
   - Used to ensure data integrity.

- Example: SHA-256

3. **Symmetric Cryptography:**
    - Occasionally used for encrypting data in conjunction with PKI

d. A Typical Application Use of PKI

- **HTTPS (Web Security):**
    - Websites use PKI to encrypt communication using SSL/TLS protocols.
    - Digital certificates authenticate the server, ensuring the client that they're communicating with the intended server.

e. Certificate Signing Request (CSR) Made by S3

Description:

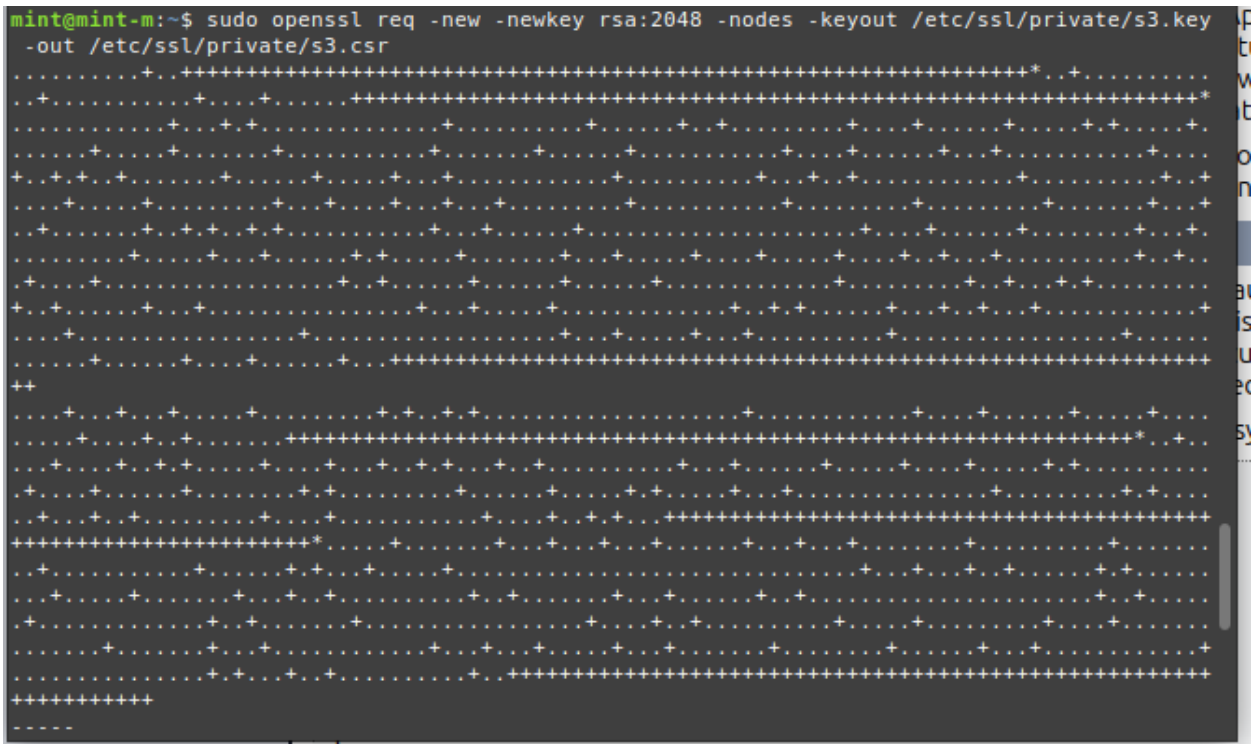
The CSR is a request sent to a Certificate Authority to issue a certificate for S3, containing:

- Public key.
- Organization details.
- Common Name (e.g., domain or IP address).

Generated CSR:

- To generate the CSR:
  sudo openssl req -new -newkey rsa:2048 -nodes -keyout /etc/ssl/private/s3.key -out /etc/ssl/private/s3.csr

  CSR:



Summary:

PKI is integral to securing communications and ensuring trust between entities. The CSR generated by S3 demonstrates its role in obtaining a trusted certificate.

Task E: Secure IP Payloads Between S1 and S4

Solution Description:

IPsec (Internet Protocol Security) was deployed to secure IP payloads between S1 and S4. It provides:

- **Confidentiality:** Encrypts the IP packets to prevent unauthorized access.
- **Authentication:** Ensures only trusted entities can communicate.
- **Integrity:** Ensures packets are not tampered with during transmission.

Implementation Methodology:

1. **Installed strongSwan on S1 and S4:**

   sudo apt install strongswan -y

2. **Configured /etc/ipsec.conf: On both S1 and S4:**

   conn S1-S4
   authby=secret
   auto=start
   left=192.168.1.1
   right=192.168.1.4
   ike=aes256-sha256-modp2048
   esp=aes256-sha256

3. **Configured /etc/ipsec.secrets:** On both S1 and S4:

192.168.1.1 192.168.1.4 : PSK "1122"

4. **Restarted strongSwan:**

   sudo systemctl restart strongswan

4. **Verified Tunnel Establishment:**
   - **Checked IPsec status:**

     sudo ipsec statusall

   - **Tested connectivity with ping:**

     ping 192.168.1.4 # From S1

     ping 192.168.1.1 # From S4

5. **Verified Encrypted Traffic:**
   - Monitored IPsec logs:

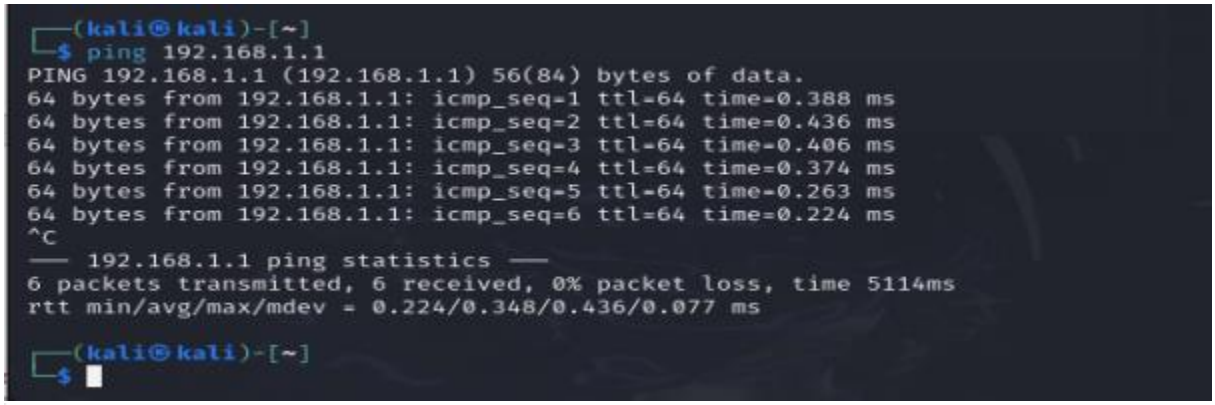
     sudo journalctl -u strongswan

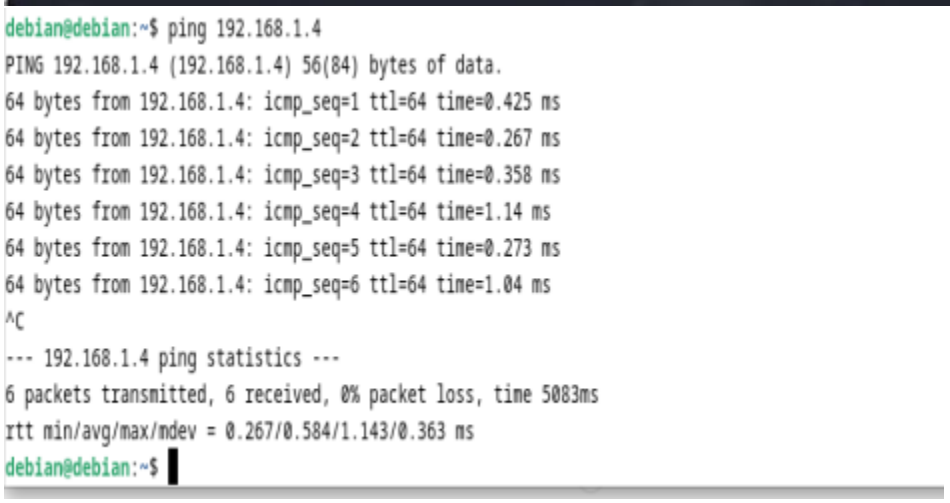
Testing Methodology:

- Verified secure communication between S1 and S4 using ping.
- Confirmed encryption by reviewing IPsec logs.

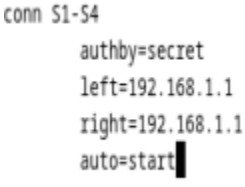
Summary:

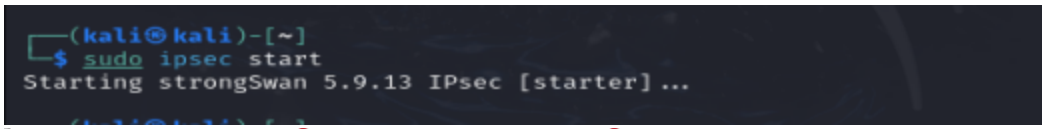
The IPsec tunnel was successfully configured to secure all communication between S1 and S4, ensuring encrypted, authenticated, and tamper-proof data transmission.





Connections established






```
debian@debian:~$ sudo ipsec start
Starting strongSwan 5.9.8 IPsec [starter]...
charon is already running (/var/run/charon.pid exists) -- skipping daemon start
starter is already running (/var/run/starter.charon.pid exists) -- no fork done
debian@debian:~$
```

```
┌──(kali㉿kali)-[~]
└─$ sudo journalctl -u strongswan
Dec 08 11:26:23 kali systemd[1]: /etc/systemd/system/strongswan.service:7: U
Dec 08 11:26:30 kali systemd[1]: Started strongswan.service - StrongSwan IPs
Dec 08 11:26:30 kali ipsec[25670]: Starting strongSwan 5.9.13 IPsec [starter
Dec 08 11:26:30 kali ipsec_starter[25670]: Starting strongSwan 5.9.13 IPsec
Dec 08 11:26:30 kali ipsec[25670]: charon is already running (/var/run/charo
Dec 08 11:26:30 kali ipsec_starter[25670]: charon is already running (/var/r
Dec 08 11:26:30 kali ipsec[25670]: starter is already running (/var/run/star
Dec 08 11:26:30 kali ipsec_starter[25670]: starter is already running (/var/
Dec 08 11:26:30 kali systemd[1]: strongswan.service: Deactivated successfull
Dec 08 11:26:30 kali systemd[1]: strongswan.service: Scheduled restart job,
Dec 08 11:26:30 kali systemd[1]: Started strongswan.service - StrongSwan IPs
Dec 08 11:26:30 kali ipsec[25676]: Starting strongSwan 5.9.13 IPsec [starter
Dec 08 11:26:30 kali ipsec_starter[25676]: Starting strongSwan 5.9.13 IPsec
Dec 08 11:26:30 kali ipsec[25676]: charon is already running (/var/run/charo
Dec 08 11:26:30 kali ipsec_starter[25676]: charon is already running (/var/r
Dec 08 11:26:30 kali ipsec_starter[25676]: starter is already running (/var/
Dec 08 11:26:30 kali ipsec[25676]: starter is already running (/var/run/star
Dec 08 11:26:30 kali systemd[1]: strongswan.service: Deactivated successfull
Dec 08 11:26:31 kali systemd[1]: strongswan.service: Scheduled restart job,
Dec 08 11:26:31 kali systemd[1]: Started strongswan.service - StrongSwan IPs
Dec 08 11:26:31 kali ipsec[25689]: Starting strongSwan 5.9.13 IPsec [starter
Dec 08 11:26:31 kali ipsec_starter[25689]: Starting strongSwan 5.9.13 IPsec
Dec 08 11:26:31 kali ipsec[25689]: charon is already running (/var/run/charo
Dec 08 11:26:31 kali ipsec_starter[25689]: charon is already running (/var/r
lines 1-24 ... skipping ...
Dec 08 11:26:23 kali systemd[1]: /etc/systemd/system/strongswan.service:7: Unkno
Dec 08 11:26:30 kali systemd[1]: Started strongswan.service - StrongSwan IPsec V
Dec 08 11:26:30 kali ipsec[25670]: Starting strongSwan 5.9.13 IPsec [starter] ...
Dec 08 11:26:30 kali ipsec_starter[25670]: Starting strongSwan 5.9.13 IPsec [sta
Dec 08 11:26:30 kali ipsec[25670]: charon is already running (/var/run/charon.pi
Dec 08 11:26:30 kali ipsec_starter[25670]: charon is already running (/var/run/c
```

```
debian@debian:~$ sudo journalctl -u strongswan
Dec 08 11:18:45 debian systemd[1]: Started strongswan.service - StrongSwan IPsec VPN Daemon.
Dec 08 11:18:45 debian ipsec[4627]: Starting strongSwan 5.9.8 IPsec [starter]...
Dec 08 11:18:45 debian ipsec_starter[4627]: Starting strongSwan 5.9.8 IPsec [starter]...
Dec 08 11:18:45 debian ipsec_starter[4627]: charon is already running (/var/run/charon.pid exists)
Dec 08 11:18:45 debian ipsec[4627]: charon is already running (/var/run/charon.pid exists) -- skip
Dec 08 11:18:45 debian ipsec[4627]: starter is already running (/var/run/starter.charon.pid exists
Dec 08 11:18:45 debian ipsec_starter[4627]: starter is already running (/var/run/starter.charon.pi
Dec 08 11:18:45 debian ipsec[4631]: Stopping strongSwan IPsec...
Dec 08 11:18:46 debian systemd[1]: strongswan.service: Deactivated successfully.
Dec 08 11:18:46 debian systemd[1]: strongswan.service: Scheduled restart job, restart counter is a
Dec 08 11:18:46 debian systemd[1]: Stopped strongswan.service - StrongSwan IPsec VPN Daemon.
Dec 08 11:18:46 debian systemd[1]: Started strongswan.service - StrongSwan IPsec VPN Daemon.
Dec 08 11:18:46 debian ipsec[4637]: Starting strongSwan 5.9.8 IPsec [starter]...
Dec 08 11:18:46 debian ipsec_starter[4637]: Starting strongSwan 5.9.8 IPsec [starter]...
Dec 08 11:18:46 debian charon[4643]: 00[DMN] Starting IKE charon daemon (strongSwan 5.9.8, Linux 6
Dec 08 11:18:46 debian ipsec[4642]: Stopping strongSwan IPsec...
Dec 08 11:18:46 debian charon[4643]: 00[LIB] providers loaded by OpenSSL: legacy default
Dec 08 11:18:46 debian charon[4643]: 00[KNL] known interfaces and IP addresses:
Dec 08 11:18:46 debian charon[4643]: 00[KNL]   lo
Dec 08 11:18:46 debian charon[4643]: 00[KNL]     127.0.0.1
Dec 08 11:18:46 debian charon[4643]: 00[KNL]     ::1
```

IPSec Secured.