Jesua Gonzalez

Ethical Hacking

Public Key Infrastructure and SSH Security

Wed Jan 21

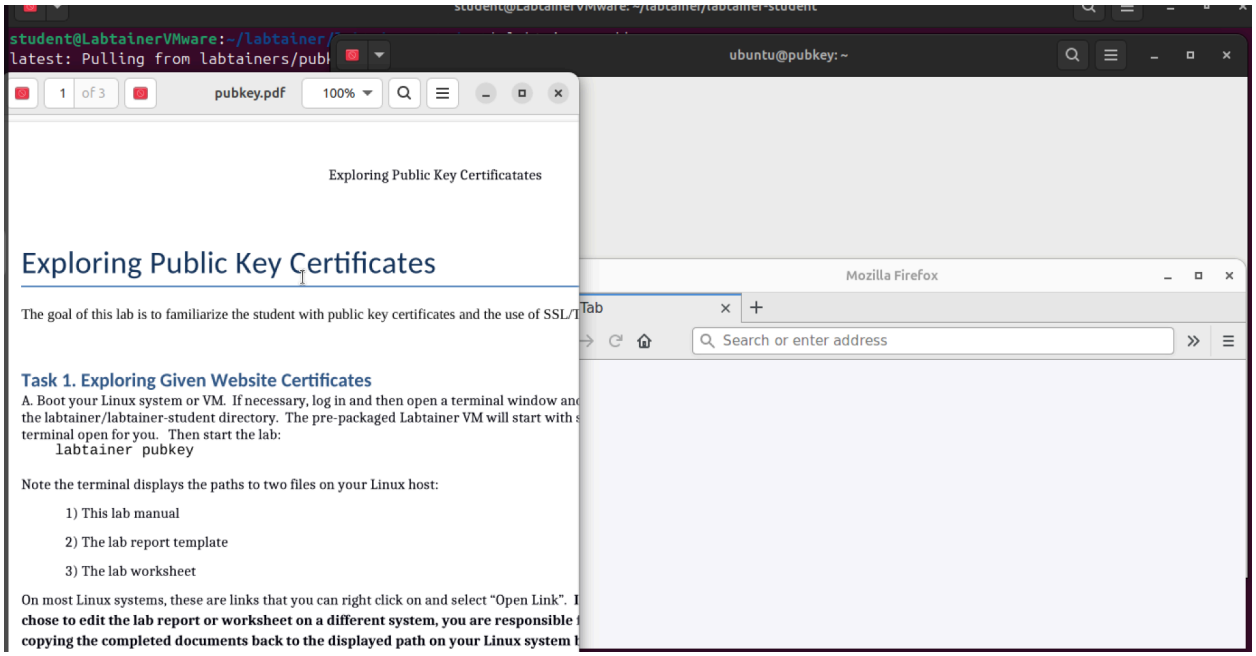
Executive Summary

This report covers my hands-on work in the Labtainers labs, where I explored Public Key Infrastructure and SSH authentication. PKI is a system that uses public and private keys plus digital certificates to establish trust on the Internet. A certificate, issued by a trusted certificate authority, proves that a public key belongs to a specific server or entity. Meanwhile, SSH authentication with keys replaces passwords with a public key installed on the server and a private key kept on the client. When you connect, the server challenges the client, and the client proves it holds the private key without sending it over the network. In the controlled lab environment, I examine digital certificates and SSL/TLS settings, generate and manage SSH key pairs, copy keys to a server, and confirm SSH logins. My goals were straightforward, I set out to show how certificates protect web traffic and why trusting the right certificate matters. I wanted to demonstrate how SSH key-based authentication removes the need for passwords and lowers the risk of credential theft.

For verification I relied on simple, commonly available tools. I used a browser and OpenSSL to read certificate fields and test TLS connections. I use ssh-keygen to create public and private keys, ssh-copy-id to install the public key on the server, and ssh to test a secure login. I recorded all outputs and commands, and included short explanations, saving them in a format that allows any reader new to cybersecurity or a non-technical executive to follow what was done and why it matters.

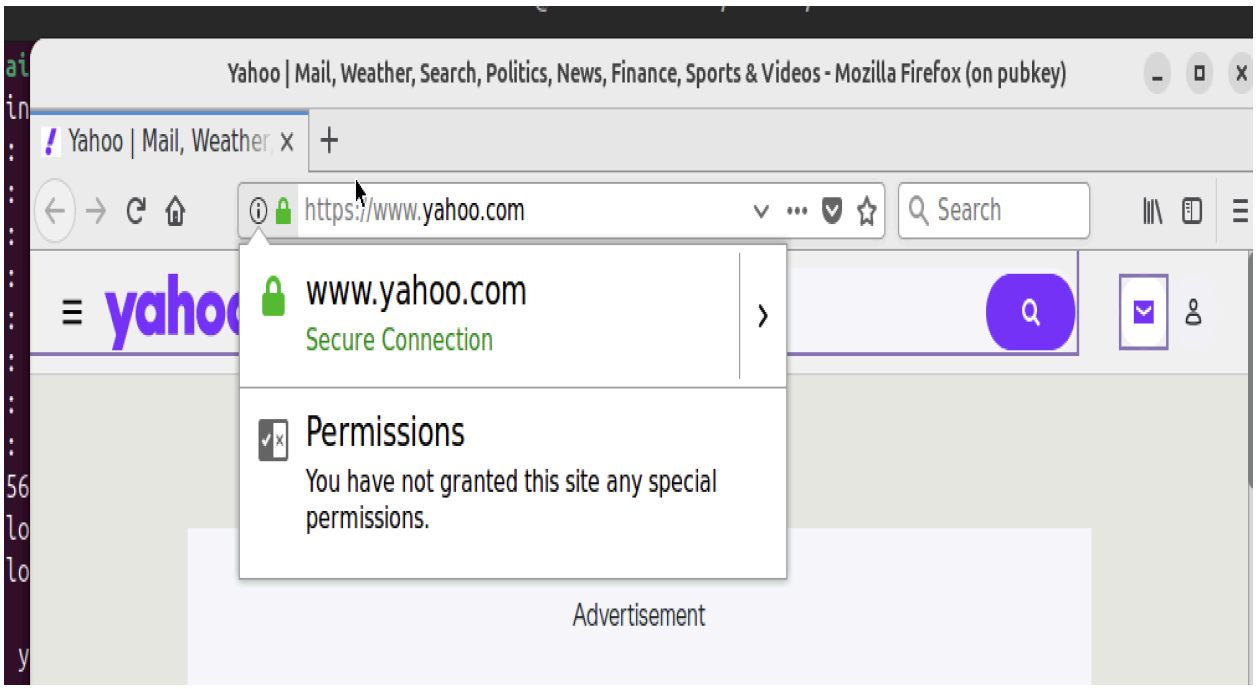
Part 1: Public Key Certificates Lab

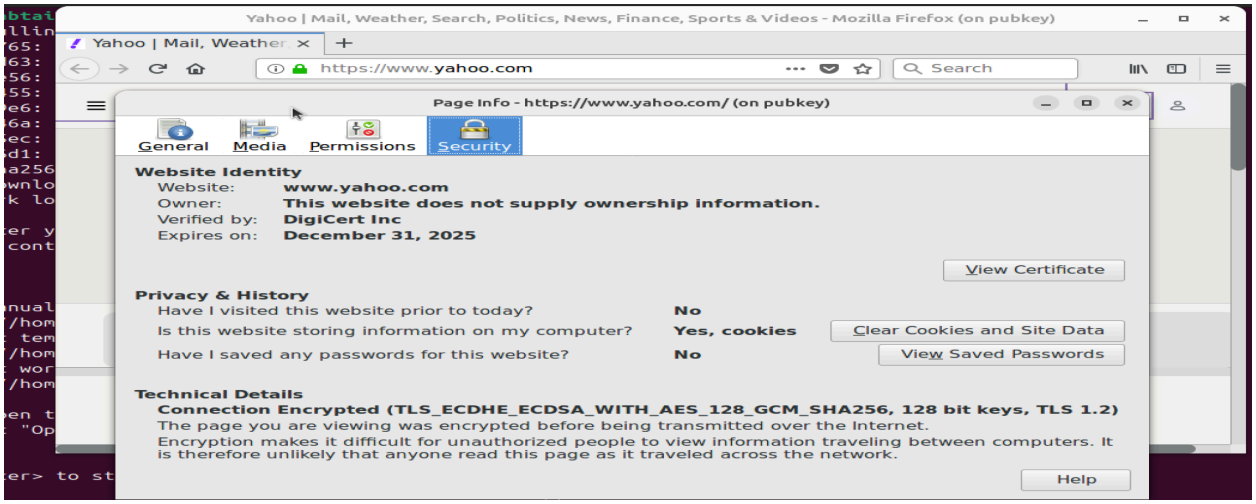
Task 0: Pubkey Lab Environment Setup

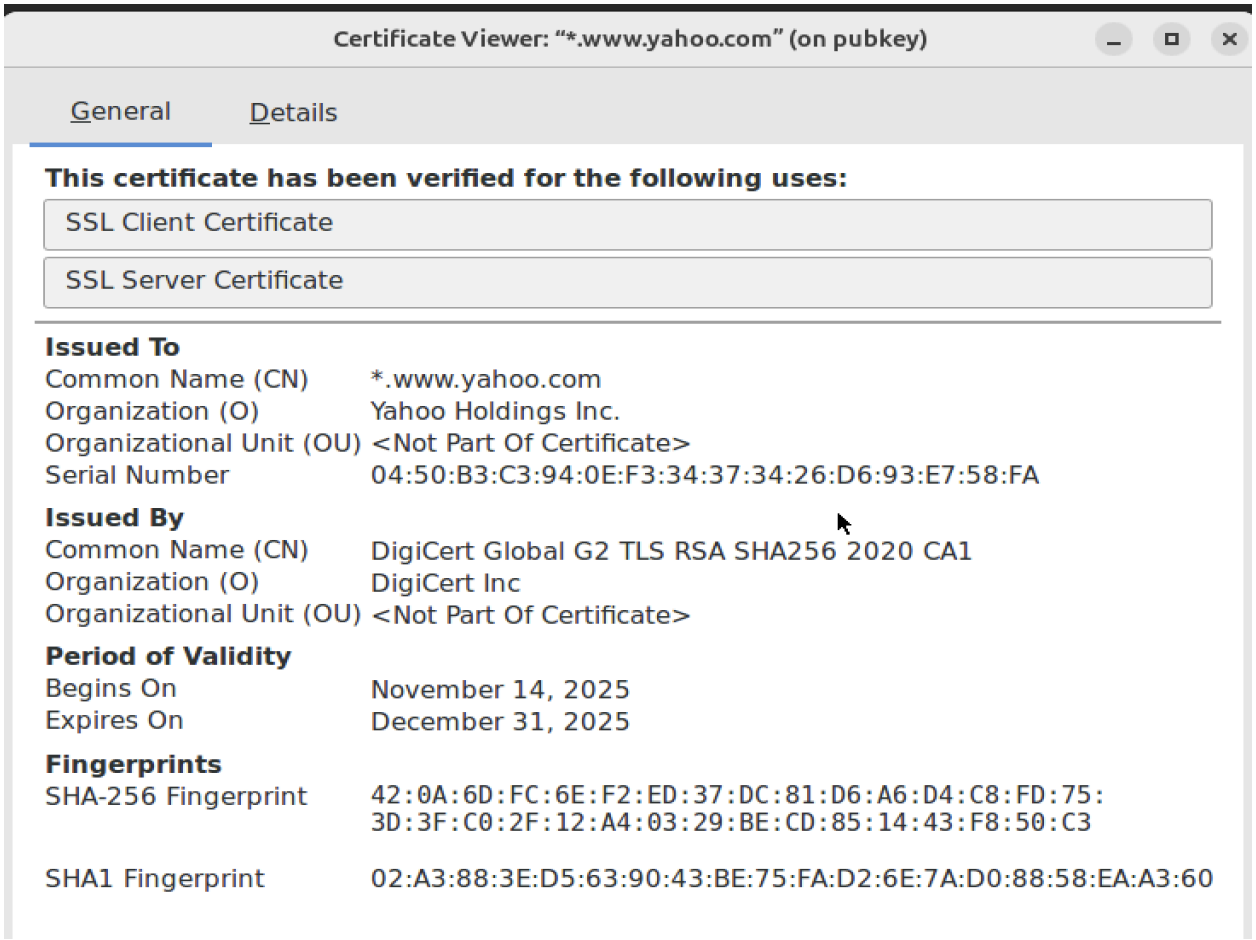


- The lab was accessed using: `labtainer pubkey`. It provides an ubuntu@pubkey terminal and a Firefox browser, as shown in the picture.

Task 1: Exploring Website Certificates

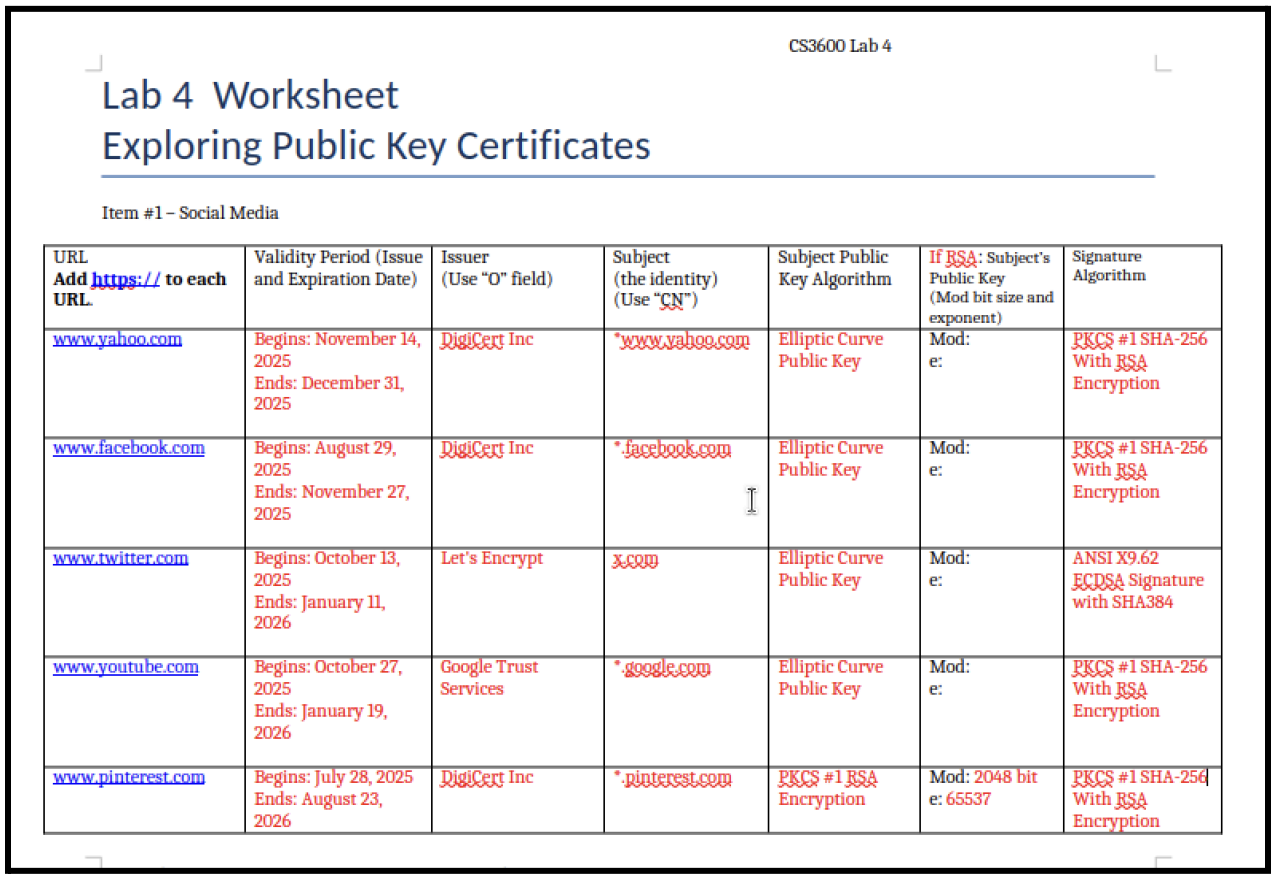






- I opened the websites in Firefox, which the lab launches for me. I then clicked the padlock next to the address bar, clicked the arrow for more options, selected "More Information", and opened the Security section. From there I can click "View Certificate" and switch to the Details tab. I inspected the Issuer or Certificate Authority, the Subject or Common Name, the validity dates for issue and expiration, the public key algorithm, the RSA key size if present, and the signature algorithm.

Task 1.1: Certificate Information Extraction

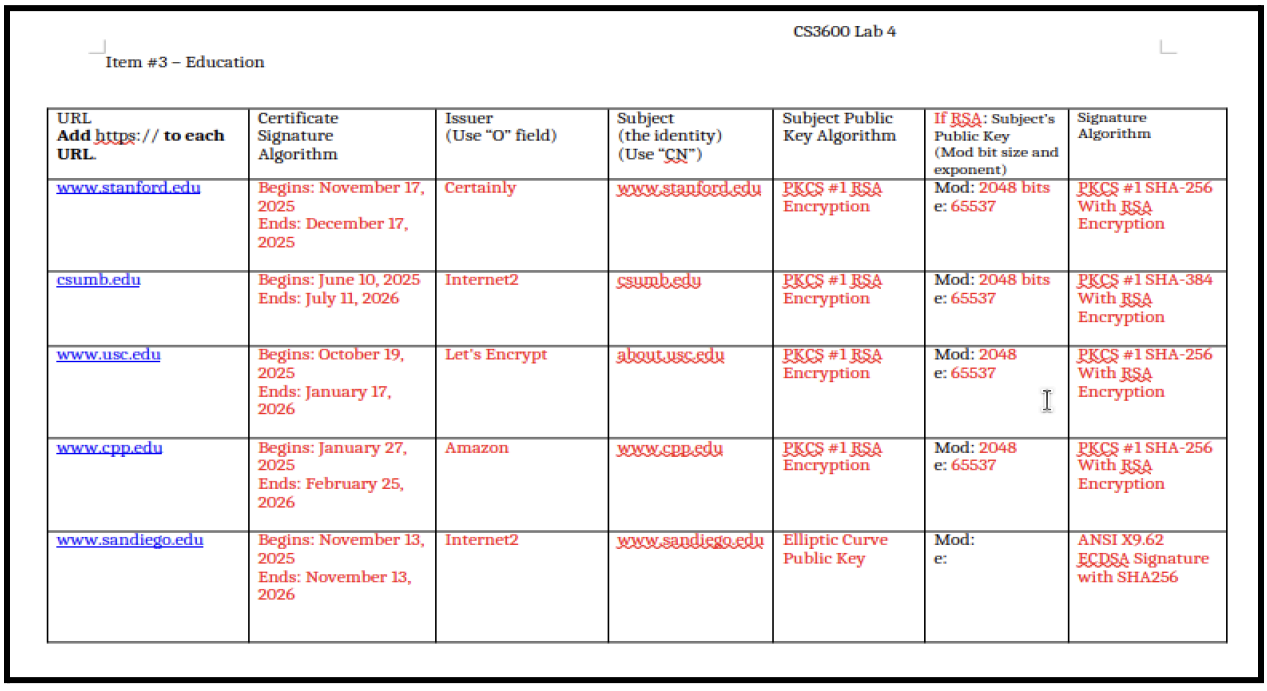


CS3600 Lab 4

Lab 4 Worksheet
Exploring Public Key Certificates

Item #1 – Social Media

URL **Add https:// to each URL.**	Validity Period (Issue and Expiration Date)	Issuer (Use "O" field)	Subject (the identity) (Use "CN")	Subject Public Key Algorithm	If RSA: Subject's Public Key (Mod bit size and exponent)	Signature Algorithm
www.yahoo.com	Begins: November 14, 2025 Ends: December 31, 2025	DigiCert Inc	*www.yahoo.com	Elliptic Curve Public Key	Mod: e:	PKCS #1 SHA-256 With RSA Encryption
www.facebook.com	Begins: August 29, 2025 Ends: November 27, 2025	DigiCert Inc	*.facebook.com	Elliptic Curve Public Key	Mod: e:	PKCS #1 SHA-256 With RSA Encryption
www.twitter.com	Begins: October 13, 2025 Ends: January 11, 2026	Let's Encrypt	x.com	Elliptic Curve Public Key	Mod: e:	ANSI X9.62 ECDSA Signature with SHA384
www.youtube.com	Begins: October 27, 2025 Ends: January 19, 2026	Google Trust Services	*.google.com	Elliptic Curve Public Key	Mod: e:	PKCS #1 SHA-256 With RSA Encryption
www.pinterest.com	Begins: July 28, 2025 Ends: August 23, 2026	DigiCert Inc	*.pinterest.com	PKCS #1 RSA Encryption	Mod: 2048 bit e: 65537	PKCS #1 SHA-256 With RSA Encryption



CS3600 Lab 4

Item #3 – Education

URL **Add https:// to each URL.**	Certificate Signature Algorithm	Issuer (Use "O" field)	Subject (the identity) (Use "CN")	Subject Public Key Algorithm	If RSA: Subject's Public Key (Mod bit size and exponent)	Signature Algorithm
www.stanford.edu	Begins: November 17, 2025 Ends: December 17, 2025	Certainly	www.stanford.edu	PKCS #1 RSA Encryption	Mod: 2048 bits e: 65537	PKCS #1 SHA-256 With RSA Encryption
csumb.edu	Begins: June 10, 2025 Ends: July 11, 2026	Internet2	csumb.edu	PKCS #1 RSA Encryption	Mod: 2048 bits e: 65537	PKCS #1 SHA-384 With RSA Encryption
www.usc.edu	Begins: October 19, 2025 Ends: January 17, 2026	Let's Encrypt	about.usc.edu	PKCS #1 RSA Encryption	Mod: 2048 e: 65537	PKCS #1 SHA-256 With RSA Encryption
www.cpp.edu	Begins: January 27, 2025 Ends: February 25, 2026	Amazon	www.cpp.edu	PKCS #1 RSA Encryption	Mod: 2048 e: 65537	PKCS #1 SHA-256 With RSA Encryption
www.sandiego.edu	Begins: November 13, 2025 Ends: November 13, 2026	Internet2	www.sandiego.edu	Elliptic Curve Public Key	Mod: e:	ANSI X9.62 ECDSA Signature with SHA256

- 5 URLs I investigated were all popular social media platforms, while the other 5 were education-related, such as university websites.

Task 2: Analysis Questions

- What are the most common certificate authorities you observed:

DigiCert showed up the most across the social media URLs I checked. Let's Encrypt also popped up several times, and for education sites I saw Internet2 and commercial issuers like Amazon and Google Trust Services, which are big well know companies.

- What key algorithms and key sizes were most prevalent:

Elliptic curve keys were common on the big social sites; many sites use EC-based keys for performance and smaller sizes. While RSA 2048 was the most common RSA mod bit size in my sample, with an exponent of 65537. The signature algorithms I recorded were mostly SHA256 with either RSA or ECDSA, and one with a stronger hash like SHA384 for csumb.com, which will produce a larger hash output.

- Why is it important for websites to use HTTPS instead of HTTP:

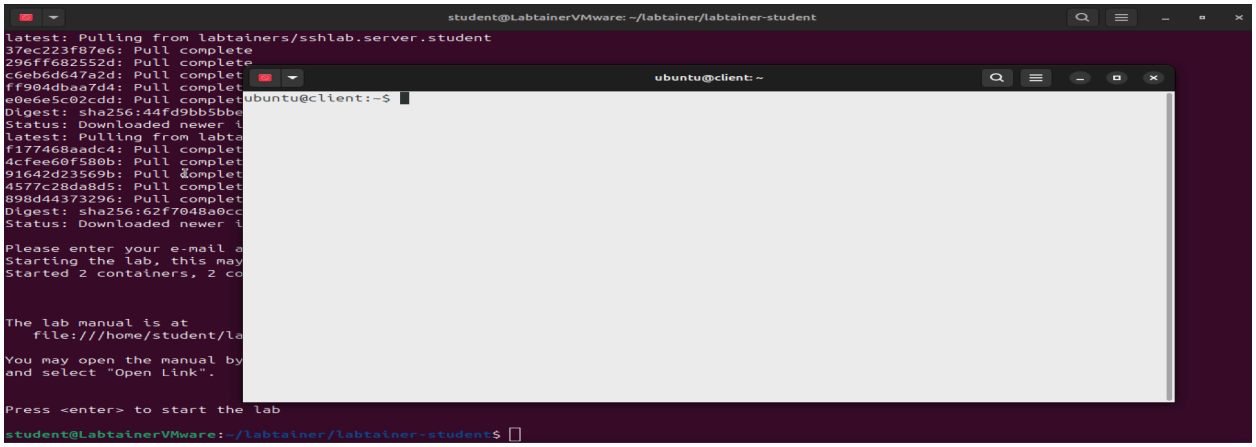
HTTPS encrypts traffic so threat actors on the network cannot read or modify data. It also proves the site's identity through a certificate, so users know they are talking to the real website. Without HTTPS credentials and private data can be stolen, pages can be changed in transit, and users can be tricked by fake sites.

- What could happen if a certificate expires or is issued by an untrusted authority:

Browsers will warn or block the site, which breaks functionality and raises trust concerns. More seriously, an expired or untrusted certificate can potentially lead to cyberattacks. Attackers may be able to perform man-in-the-middle attacks more easily, exposing passwords and sensitive data. It can also cause regulatory and business consequences if protected data is leaked or customers lose confidence and security.

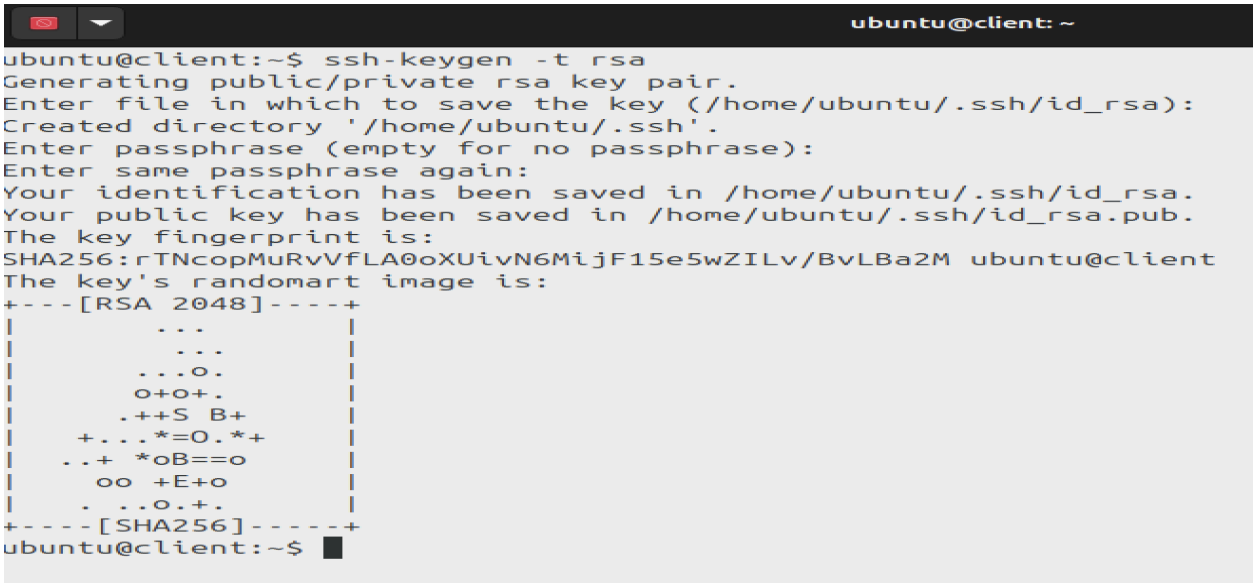
Part 2: SSH Security Lab

Task 0: SSH Lab Environment Setup

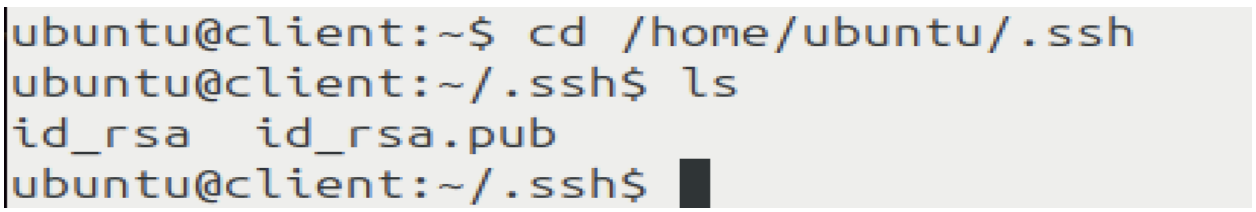


- The lab was accessed using: `labtainer sshlab`. It provides an ubuntu@client terminal as shown in the picture.

Task 1: Generate SSH Key Pair



- I generated an RSA key pair on the client with `ssh-keygen -t rsa`. I then check my ~/.ssh directory and see the files id_rsa and id_rsa.pub, which are the private and public keys.



Task 1.2: Analysis Questions

- What is the difference between the private key and public key:

A public key is meant to be shared so others can verify you or encrypt data for you. The private key is secret. You keep it on your device and use it to prove your identity or decrypt what was encrypted for you.

- Why must the private key be kept secure:

If someone gets your private key, they can impersonate you or log into servers and access any systems that trust that key. An attacker can bypass passwords and cause large-scale damage.

- In production, why would you use a passphrase to protect the private key:

A passphrase encrypts the private key file on disk. If the file is stolen the attacker still needs the passphrase to use it. This adds an extra layer of defense and can reduce risk.

Task 2: Configure SSH Server and Connect

```
ubuntu@client:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub 172.20.0.3

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_rsa.pub"
The authenticity of host '172.20.0.3 (172.20.0.3)' can't be established.
ECDSA key fingerprint is SHA256:nFDnpYXdisAGpF1ZxOBv8Xc83CDp5qYU2frYQvB7Pt8.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
 already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
ubuntu@172.20.0.3's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.20.0.3'"
and check to make sure that only the key(s) you wanted were added.

ubuntu@client:~$
```

- I copied my public key to the server with `ssh-copy-id -i ~/.ssh/id_rsa.pub 172.20.0.3`. This adds my RSA key I just made so that I can log in to the server without a password and instead authenticate using my encrypted credentials.

Task 2.2: Connect via SSH

```
ubuntu@client:~$ ssh ubuntu@172.20.0.3
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 6.14.0-33-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
ubuntu@server:~$ cat filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (sshlab-server) container

# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: 3342738a9f706912e7bcafa031181ea5
ubuntu@server:~$ exit
logout
Connection to 172.20.0.3 closed.
ubuntu@client:~$
```

- I connected to the server with `ssh ubuntu@172.20.0.3`. Since I copied my public key I can log in without a password prompt. Once connected, I ran `cat filetoview.txt` to display the file contents as proof of access.

Task 2.3: Analysis Questions

- What does the ssh-copy-id command do:

It connects to the remote account, creates the .ssh directory if needed, and adds your public key to the remote authorized keys file.

- Where is your public key stored on the server:

It is stored in the user's SSH folder at /home/ubuntu/.ssh/authorized_keys

- Explain why you were not prompted for a password when connecting:

My device was able to confirm my identity since the server found my public key and was able to communicate with it by sending a challenge that my private key was able to complete.

- Compare key-based authentication to password authentication: what are the advantages:

Key-based authentication holds advantages like being able to restrict or revoke keys, as well as being harder to brute force or guess easily. Password authentication on the other hand, is a simpler process and does not require a long distribution process.

Task 3: Integration and Reflection

Connection Between PKI and SSH - Each method has a public key that can be shared and a private key that must stay secret. In PKI the public key lives in a certificate signed by a

certificate authority, and that signature creates a chain of trust that browsers and hosts validate. In SSH the trust model is different; both servers and hosts learn and store server keys. That means PKI relies on third-party trust by default while SSH usually relies on direct trust between the two endpoints. Functionally, they are very similar; in both methods the public key is used to verify identity or to encrypt session keys, and the private key is used to prove identity or decrypt. Both create a way to authenticate without sending passwords over the network.

Real-World Applications - SSL/TLS certificates protect any web service that handles sensitive data. For example, online banking and e-commerce checkout pages have payment pages and personal information that must stay private. They are also essential in webmail services where encrypting traffic and proving server identity prevents compromise. SSH key-based auth is the standard for server administration and cyber folks, giving us secure & repeatable access to machines. It is also heavily used in automated deployments and CI/CD pipelines where agents need passwordless but controlled access to cloud VMs and build servers.

Personal Reflection - The biggest thing I learned is how public and private keys prove identity and then establish a secure session, and the labs made that idea concrete. In the PKI lab I inspected certificates in the browser, checked who issued them and their validity periods, and recorded the fields that show trust. In the SSH lab I generated key pairs, used ssh-copy-id to install my public key, and logged in without a password, which demonstrated key-based authentication end to end. I will use this knowledge by enforcing trusted certificates for any service I manage and by using SSH keys for admin access and automation while rotating keys and protecting private keys with passphrases. A question I still have is how much computing power will one need to crack different encryption hashes like RSA or Elliptic Curve?

References

- Badman, Annie, and Matthew Kosinski. "What Is Asymmetric Encryption?" *IBM Think*, IBM, n.d., https://www.ibm.com/think/topics/asymmetric-encryption. Accessed 19 Nov. 2025.

- National Institute of Standards and Technology. “Public Key Infrastructure.” *NIST Computer Security Resource Center*, https://csrc.nist.gov/glossary/term/public_key_infrastructure. Accessed 19 Nov. 2025.

- Ellingwood, Justin. "Understanding the SSH Encryption and Connection Process." *DigitalOcean*, 1 Apr. 2022, https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process. Accessed 19 Nov. 2025.